La conception et l'ingénierie malgré l'open-source

Aurélien Pierre

MANUSCRIT, VERSION 2

Édition en accès libre — première version : 2 août 2026

Dernière mise à jour : 28 août 2026

editions.aurelienpierre.com/concevoir/

Un outil sert son utilisateur, ou il le trahit. Il n'y a pas d'entre-deux.

Ce livre fait deux choses. Il documente — registres ouverts, tout compté, daté, vérifiable — la façon dont un des logiciels libres les plus connus de la photographie traite ses utilisateurs, ses contributeurs et ses critiques. Et il expose la méthode dont ce projet ne voulait pas : poser le problème avant la solution, écouter sans obéir, mesurer sans se mentir, refuser ce qu'on ne saura pas maintenir.

J'y étais. J'ai écrit une partie de ses modules, je l'ai quitté, je l'ai forké. Tout ce qui pouvait se mesurer l'a été ; le reste est donné pour ce que c'est : mon jugement, signé.

La conclusion emploie un mot que l'ingénierie ne prononce jamais : concevoir, c'est prendre soin. Et le logiciel libre, qui a confié les outils de tous à ceux qui s'amusent, ne peut pas rester l'amusement de quelques-uns qui peuvent, au détriment de la majorité qui a besoin.

Prologue — Trois chemins

La rédaction de ce livre a emprunté 3 chemins différents pour arriver au résultat que vous avez sous les yeux. Le premier chemin était de compiler, synthétiser et approfondir une quinzaine de mes articles écrits sur la conception, et 50 minutes de notes vocales, agrégés au cours de la dernière décennie et ancrés dans mon expérience de recherche et développement dans le logiciel libre et open-source, et d'utiliser cette occasion pour enrichir ou nuancer mes observations empiriques par ce que la littérature scientifique avait à dire des mêmes sujets. Les concepteurs, même non-ingénieurs en général, et non-programmeurs en particulier, devraient donc trouver dans les premières parties des méthodes d'analyse des problèmes et d'identification de solutions qui s'appliquent à de nombreux domaines.

Le chemin suivant a été de former mon successeur à la maintenance du projet Ansel, un projet de logiciel de traitement photo que j'ai forké du projet Darktable en 2022, à la suite d'une énième refonte massive de l'interface utilisateur, faite au détriment de l'utilisabilité et sans besoin ou problème défini, et qui signait pour moi la fracture définitive entre l'équipe de développement et mon éthique de travail. Depuis longtemps, je disais que si j'avais un jour à donner un cours de conception, le projet Darktable serait mon cas d'école de tous les exemples à ne pas reproduire, tant les mauvaises décisions ont été prises partout

où il était possible de les prendre. Ce jugement se mesure, et ce livre le mesure sur quatre registres : l'assistance aux utilisateurs et leur engagement, qui s'effondrent ; les contributions, qui déclinent ; la correction des anomalies, qui ne suit plus ; la complexité du code, chiffrée aux métriques standard, qui s'envole. Et ces décisions ne furent pas des paris malchanceux : sortir à date fixe sans salariés pour encaisser le rythme, fusionner plus vite qu'un humain ne peut relire, accepter sans critères, empiler les options plutôt que trancher, laisser le pouvoir se concentrer sans mandat ni appel — autant de choix que la littérature, celle du génie logiciel comme celle des organisations, avait déjà classés parmi les mauvaises idées, des décennies avant qu'ils ne soient pris. Il est donc important de documenter ces erreurs, par leurs effets autant que par ce mépris des connaissances établies, pour ne pas se condamner à les reproduire. Elles sont à la fois le fruit d'une méconnaissance de ce que je vais présenter ici (méthodes de conception industrielle et structures psycho-sociales de l'appareil productif), et d'une absence d'intérêt pour l'Autre, c'est-à-dire celui qui n'est pas « Nous » (la typologie sociale de ce « Nous » et ses codes seront détaillés). Il est important de comprendre finement pourquoi ces erreurs ont été des erreurs, et pas seulement des accidents, dans quelles strates de l'appareil de production elles se sont produites, car utiliser la stratégie diamétralement opposée, aveuglément, ne ferait que générer un type différent d'erreur : l'espace des mauvaises solutions est infini. Je ne souhaite pas qu'Ansel devienne un autre Darktable.

Le dernier chemin est le récit d'une solitude qui se transforme en plaidoyer : celle d'un développeur qui programmait pour régler

les problèmes des humains devant la machine, au milieu d'une communauté qui programme parce que c'est amusant, et qui s'enthousiasme indifféremment pour toute innovation technologique, sans jamais se demander si cette nouveauté va rendre son travail meilleur et plus facile. Pire, elle ne se demande jamais en quoi ce qui amuse chez Nous peut devenir une souffrance chez l'Autre. Car cette communauté se définit en premier par son éducation et par ses privilèges, et donc par les besoins qu'elle a le luxe d'ignorer, tout en entretenant un double discours d'émancipation et de souveraineté numérique qui ne sert qu'à réduire les besoins (et les plaintes) de l'utilisateur au silence.

La conclusion de ce livre s'écrivait d'elle-même, chapitre après chapitre, sans que je ne m'en aperçoive : la conception bien faite, l'ingénierie au service de l'humain, se doit de retirer à la médecine l'exclusivité du soin. Confiner l'ingénierie aux problèmes techniques est une myopie, car il n'y a pas de problème technique sans humain *technicisé* démuni devant une machine, ou devant un problème d'une ampleur telle que la productivité de la machine devient la seule voie réaliste. Le technicisé est un humain dont il faut prendre soin, parce que la technique peut autant soulager des souffrances qu'en créer de nouvelles, et des pires. On n'hésite plus à présenter la responsabilité de l'ingénieur en termes d'impact social et environnemental, mais ce sont des concepts larges, des objets sans forme qui n'engagent qu'à produire des métriques vagues prouvant qu'on a étudié la question, et puis à continuer tout pareil en se faisant croire que le progrès est un train qui avance

mû par sa nature propre et inévitable, qu'on choisisse d'y monter ou pas. Non, le soin, c'est le rapport de celui qui peut sur celui qui a besoin, et cette asymétrie crée le pouvoir qu'on essaie de cacher sous l'apparence technique du problème, car c'est au fond un rapport de domination d'un humain sur un autre. Rien de moins, rien de plus.

Et c'est précisément ce qui ne va pas dans le logiciel libre : un cercle de privilégiés qui fabrique des outils pour ce même cercle, en prétendant produire des communs universels, via des concepts vaguement libertaires et faussement marxistes, qui ne servent qu'à convaincre les exclus du cercle qu'ils ne doivent leur exclusion qu'à leurs caractéristiques personnelles, à leurs défauts internes. Quand c'est la structure du système qui produit elle-même ses exclus.

Mais ce soin, pour être juste, ne saurait ignorer celui qui engage sa force de travail, son temps et son énergie, au service des besoins des autres ; et il ne s'agit pas non plus de le transformer en oppression et en exploitation de celui qui doit produire par ceux qui veulent utiliser, d'autant que les producteurs y sont toujours plus rares que les utilisateurs, et que la foule est une masse dont il faut aussi se protéger. On pourrait être tenté d'y voir un équilibrage du rapport de force, mais c'est seulement un cercle vicieux qui ne fait que légitimer une violence à double sens. Or c'est un autre travers du logiciel libre, qui trouve en réalité sa source dans le même problème : penser la structure sociale dans laquelle le logiciel libre émerge est l'angle mort du mouvement libre, qui a focalisé toute son attention sur les

problèmes de licences, c'est-à-dire de propriété intellectuelle, et rien d'autre.

La thèse de ce livre tient en 4 points :

1. La conception est une discipline de résolution d'un problème humain, dont on a choisi de faire une lecture technique — et l'objet de la conception n'est jamais seulement la machine, mais le système entier : le moteur, l'interface, la documentation, la communauté, l'économie qui la porte, et le travailleur qui maintient le tout.
2. Le logiciel libre doit cesser d'être le terrain de jeu d'un club socialement homogène pour devenir un service public qui répond aux besoins de ses usagers avec bienveillance, d'autant plus que sa rhétorique joue déjà sur les deux tableaux.
3. La conception dans un service public devient alors un acte de soin, et le soin induit une relation asymétrique qui requiert une déontologie stricte pour ne pas devenir un acte d'oppression dans une structure de domination.
4. La garantie du respect de la déontologie, l'arbitrage entre les besoins des usagers, entre les besoins des travailleurs, et la garantie de la sécurité des conditions de travail ne peuvent pas s'envisager hors d'une structure sociale organisée, qui identifie correctement les positions de pouvoir, leur demande des comptes, choisit les individus qu'elle y place et pour combien de temps.

Ces trois chemins ont laissé un livre à deux sources. La première est la littérature : ce livre emprunte énormément. Il n'invente

aucune loi : il applique à un domaine qui leur avait échappé — la conception des logiciels libres — des résultats établis ailleurs, souvent de longue date.

En filigrane de tout, l'**épistémologie**, parce qu'elle pose les questions cruciales et en propose déjà des réponses : quelle est la différence entre la connaissance et la croyance, comment réconcilier les subjectivités avec l'objectivité attendue ; comment sait-on ce qu'on sait, et quel crédit y accorder. Les réponses toutes prêtes qu'on réutilisera sont la reproductibilité des résultats — la norme même de la méthode expérimentale —, les critères de falsification de Popper, l'intégrité que Feynman opposait à la « science du culte du cargo », et la part *tacite* du savoir que Polanyi a nommée, celle qu'aucune règle ne peut formaliser.

Pour la démarche de conception, on s'appuie sur les **sciences de la conception** — Simon, pour qui concevoir c'est transformer une situation existante en situation préférée ; Rittel et Cross, pour ce qui sépare un problème de conception d'un problème ordinaire. L'**ingénierie** y ajoute ses méthodes, éprouvées sur les objets physiques : la conception systématique de Pahl et Beitz, la sélection de concepts de Pugh, le développement de produit d'Ulrich et Eppinger ; la culture **industrielle de la qualité et de l'amélioration continue** — l'analyse de la cause racine, le diagramme d'Ishikawa, les « cinq pourquoi » de Toyota, l'élimination du gaspillage du *lean* ; et l'**ingénierie de la fiabilité et de la sécurité**, qui apprend à concevoir pour que les choses tiennent et échouent proprement (Reason, Leveson, Dekker, Hollnagel).

Du côté de l'usager, ce sont les **facteurs humains, l'ergonomie et l'interaction humain-machine** qui donnent leurs lois : l'utilisateur moyen qui n'existe pas (l'anthropométrie de Daniels), le coût du choix (Hick), l'erreur qui est presque toujours une erreur de conception (Norman), l'enquête contextuelle qui va voir le travail réel au lieu de le supposer. Et pour comprendre comment cet usager perçoit, choisit et se trompe — et ce qu'est au juste la compétence —, la **psychologie cognitive et comportementale** (de White et McClelland aux biais de la décision) et la **métacognition**, cette discipline de penser contre soi.

Pour le collectif — le pouvoir qui se dissimule là où l'on croit l'avoir aboli, le silence qui s'organise, la médiocrité qui s'installe —, on emprunte à la **sociologie du travail et des organisations** : la loi d'airain de Michels, la tyrannie de l'absence de structure de Freeman, le silence organisationnel, l'oblitération par incorporation de Merton, l'étiquetage de Becker. Pour ce que le milieu fabrique, et pour qui, à la **sociologie et à la philosophie des techniques** : l'*I-methodology* d'Akrich, l'habitus de Bourdieu, la distinction de Mumford entre l'outil et la machine, la critique de la technique d'Ellul.

Pour l'économie de ce monde sans prix, à l'**économie du libre et des communs** (Lerner et Tirole, Eghbal, Hirschman, Ostrom) et à l'**économie comportementale et à la théorie de la décision** : le sophisme du coût irrécupérable, l'aversion à la perte de Kahneman et Tversky, les lois de Goodhart et de Campbell sur les mesures qui deviennent des cibles, les valeurs protégées qu'on refuse de mettre en balance.

Pour l'inclusion de l'utilisateur dans la chaîne de conception, à l'**éthique du *care*** — Tronto, Fisher, Gilligan — et à la philosophie de la convivialité d'Illich. Et pour l'analyse des procédés de fermeture et de censure — comment un désaccord de fond se requalifie en faute de ton, comment la colère se discrédite en hystérie —, à la **philosophie féministe et à la théorie critique** : l'injustice épistémique de Fricker, la parole d'Audre Lorde sur les outils du maître.

Voilà la première source de ce livre : la littérature, ce que d'autres ont établi ailleurs, sur d'autres terrains. Il en a une seconde, d'une tout autre nature — le lecteur la connaît déjà : le cas Darktable, celui du deuxième chemin. La théorie énonce des lois ; le cas Darktable montre ce qu'il en coûte de les violer de manière répétée. Cette source n'a rien de la première, car elle n'est ni neutre ni objective, mais tout ce que j'affirme est vérifiable, pour s'être déroulé en public, dans des lieux qui ont laissé des archives.

Un cas pareil vaut par deux choses, et la seconde pèse plus que la première. La première : il n'en existe pas de mieux documenté. Dix ans d'historique du dépôt de code, deux forums interrogeables par leur interface de programmation, le journal d'un salon de discussion, un traqueur d'anomalies, mes propres enquêtes d'utilisateurs publiées avec leurs données et leur code — tout y est public, et chaque affirmation, des deux camps, peut être confrontée à l'enregistrement. La seconde chose est celle qu'un étranger, les mêmes archives sous les yeux, n'aurait pas : je sais où regarder, et comment lire. Je sais quel fil compte, quel *commit* a tout fait basculer, ce qu'un silence sur un canal veut dire, quelle décision anodine en masquait une autre. Les archives sont

ouvertes à tous ; leur sens ne l'est qu'à qui a vécu dedans. C'est là ce que j'apporte : un savoir situé, que sa situation même rend à la fois plus fin et moins neutre.

Une source pareille — empirique, partiale, intéressée — n'est utilisable qu'à une condition : que la méthode qui la traite soit dite. Elle tient en 4 règles. *La littérature d'abord* : chaque mécanisme invoqué a été décrit des décennies avant ce cas, sur d'autres terrains ; le cas illustre des lois connues, il n'en fonde aucune. *La falsification* : ce que la communauté affirme, je le teste contre les registres, et je publie le résultat dans les deux sens : les mesures qui remontent figurent dans les mêmes graphiques que celles qui tombent, et chaque légende porte ses propres réserves. *La symétrie* : mes messages, mes signalements et mes commits sont exclus des comptes chaque fois que la question porte sur l'état de la maison et non sur le mien ; et ma part du conflit est versée au dossier, datée, avec les mêmes instruments que celle des autres. *Les rôles* : hormis les auteurs publics cités pour leurs écrits publics, chacun apparaît ici par sa fonction — le mainteneur, le modérateur, l'auteur d'une refonte —, jamais pour être désigné.

Tout ce que j'avance de ce cas, je l'appelle *le dossier*. Le mot ne désigne pas une archive privée : le dossier, c'est l'ensemble des pièces citées en note au fil des chapitres — fils de forum, rapports d'anomalie, historique du code, journaux de salon —, chacune datée et sourcée, et, pour tout ce que le web permet d'archiver, sauvegardée sur une archive publique à la date de consultation. Le cas est borné à dessein — il occupe les parties « Le collectif » et « La liberté et ses conditions », et les chapitres de méthode s'appuient sur la littérature et sur des exemples que chacun peut

refaire. Sa fonction, elle, traverse tout le livre. Ce qui est arrivé là n'est pas un accident de personnes : c'est la trajectoire naturelle d'un projet porté par l'habitus, les croyances et l'éthos ordinaires de cet écosystème — et la plupart des projets libres ressemblent d'avantage à celui-ci, pas aux géants salariés qu'étudie la littérature. Le livre montre ce que ces croyances coûtent, compte les dégâts sur registres publics, et défend en face des valeurs qui placent l'usager au centre. Je documente cette trajectoire pièce par pièce pour qu'Ansel ne la reproduise pas, et pour que tout projet qui veut l'éviter puisse la reconnaître à temps.

Le logiciel libre est un milieu de travail dont personne ne recrute l'équipe, où nul ne répond de ce qu'il livre. Quand une décision fait des dégâts, il n'existe ni hiérarchie à saisir, ni évaluation, ni licenciement — il ne reste que la discussion publique, qui s'envenime précisément parce qu'elle est le seul levier. J'ai usé de ce levier : ce qui devait être dit l'a été, dans les termes les plus exacts, pendant que je subissais les décisions et que j'en nettoyais les dégâts ; ma colère était légitime et fondée, et réduire un discours à sa forme pour en invalider la substance est précisément l'un des mécanismes que nous étudierons. Ce livre en tire la conclusion utile : quand un petit nombre garde des années durant la main sur les mêmes décisions, sans que personne les ait mandatés ni puisse les révoquer, le problème dépasse les individus — c'est le système qui rend le verrouillage possible.

Le contrat de lecture se déduit de tout ce qui précède. Ce livre est opinioné : je crois qu'on peut dire d'un design qu'il est bon ou mauvais, et je le dirai. Ce qui est mesuré ou établi porte sa source

en note ; tout le reste est mon jugement, écrit à la première personne. Le livre n'en est pas moins falsifiable : lorsque la littérature contredit ma position sur le même objet, elle est citée avant d'être contredite.

Ce livre, enfin, ne parle pas que de logiciel : il parle d'outils de travail et d'outils de création artistique — ces objets conçus pour servir l'intention et le but de quelqu'un d'autre. Le logiciel en est le cas d'étude principal, parce que c'est mon terrain et parce que ses interfaces fluides donnent au concepteur plus de degrés de liberté qu'aucun objet physique — assez de liberté pour faire très bien, et très mal. Mais un pinceau, un pont ou un formulaire administratif obéissent aux mêmes lois : ils servent l'intention de leur utilisateur, ou ils la trahissent.

Partie I — Ce que concevoir veut dire

1. Le mot et la chose

Le français a fait au mot *design* un sort curieux : il l'a emprunté à l'anglais en ne gardant que sa troisième acception, la moins intéressante. Chez nous, le design est affaire de formes — un canapé est *design*, une bouilloire a du *design* —, et l'on récite volontiers que le design, c'est « quand la forme rencontre la fonction ». L'anglais dit autre chose. *Design* y descend du *disegno* des ateliers italiens, qui désignait à la fois le dessin et le dessein : le tracé et l'intention. *To design*, c'est projeter — arrêter un but, puis organiser les moyens de l'atteindre. Ce que l'anglais appelle *design*, le français de l'ingénieur l'appelle **conception**[1], et c'est de cela que ce livre traite. Le dessin des formes n'en est qu'une conséquence lointaine, et pas la plus importante.

Cette confusion de vocabulaire n'est pas un détail d'étymologie : elle décide de qui se sent autorisé à parler. Tant que le design est affaire de formes, tout le monde a des yeux, donc tout le monde a un avis, et tous les avis se valent. Dès que le design est une conception — un problème posé, des contraintes, des critères, une solution qui se défend —, l'avis cesse de suffire : il faut des raisons. La plupart des discussions stériles sur le design d'un logiciel tiennent dans ce glissement de sens, et le premier travail est donc de fixer le mot.

Herbert Simon a donné de la chose la définition la plus large et la plus utile : conçoit « quiconque élabore des plans d'action visant à changer des situations existantes en situations préférées »[2]. Chacun de ses termes est important. *Des situations existantes* : la conception part d'un état du monde, pas d'une page blanche ni d'une envie. *Préférées* : il y a un but, donc quelqu'un qui préfère — un humain dont l'intention fait la mesure de la réussite. *Des plans d'action* : la conception produit un chemin organisé, pas une inspiration. Sous cette définition, l'ingénieur conçoit, mais aussi le médecin qui bâtit un protocole, le législateur qui rédige une loi, l'artisan qui adapte son outil. La conception est le cœur historique du métier d'ingénieur — même si l'ingénieur contemporain passe l'essentiel de son temps entre réunions et tableurs, son premier métier, fondamentalement, est celui-là.

Le contresens le plus coûteux : le design serait un art. L'idée est flatteuse et elle est partout — dans les écoles, dans les studios, dans les discussions de projet. Elle est surtout commode, car si le design est un art, alors tout le monde a raison, donc personne n'a tort, et il n'existe ni bonne ni mauvaise réponse : chaque proposition vaut chaque autre, et le débat est infini par construction. Ce relativisme n'est pas un garde-fou contre l'arrogance des experts ; c'est une machine à discussions sans fin. Les théoriciens de la planification Horst Rittel et Melvin Webber ont décrit dès 1973 cette propriété des problèmes de conception, qu'ils rangeaient parmi les « problèmes retors » (*wicked problems*) : ces problèmes n'ont **pas de règle d'arrêt**[3]. Rien, dans la nature d'un problème de conception, n'indique que la discussion est terminée ni que la solution est atteinte — on peut

toujours reformuler, toujours objecter, toujours préférer autre chose. La conclusion pratique est immédiate : puisque la règle d'arrêt ne viendra pas du problème, elle doit venir de la méthode. Des critères de validation posés avant la solution sont ce qui permet un jour de dire : c'est terminé, et voici pourquoi. Un design dont personne ne peut dire pourquoi il est bon — contre quels critères, sous quelles contraintes, au service de quel but — n'est pas un design : c'est une préférence qui a trouvé un budget. C'est aussi ce qui rend fausse la formule flatteuse selon laquelle « tout le monde est concepteur » : chacun peut bricoler sa table de chevet à son goût, personne n'improvise un outil dont dépendront des milliers d'inconnus — l'intuition suffit à la première, jamais au second.

Le symétrique est tout aussi faux : le design n'est pas une science. La science décrit ce qui est ; la conception décide de ce qui sera, sous contraintes, pour quelqu'un. Elle mobilise des connaissances scientifiques — sans les mathématiques et la physique, on ne borne même pas l'espace des solutions possibles —, mais elle ne s'y réduit pas : aucune équation ne choisit à votre place entre deux architectures valides, et le choix engage des humains que les équations ne décrivent pas. Nigel Cross a défendu l'idée que les manières de connaître propres à la conception forment une **troisième discipline**, distincte des sciences comme des humanités, avec ses objets et ses méthodes[4]. C'est la position de ce livre : le design se juge — donc il n'est pas un art au sens du relativisme ; il se juge *sur des critères qu'il faut construire* — donc il n'est pas une science. Le design est une discipline.

Il y a pourtant une chose que la conception doit emprunter aux sciences expérimentales : la falsifiabilité. Popper en avait fait son critère de démarcation — une théorie n'est scientifique que si elle énonce les conditions sous lesquelles elle serait reconnue fausse[5]. Transposons : un design n'est un design que s'il énonce les conditions sous lesquelles il aura échoué — ses critères, posés avant, vérifiables après. Non pour faire de la conception une science, mais pour deux raisons qui n'ont rien d'académique.

La première : l'objet conçu servira un public plus large que le panel d'essai — des milliers d'inconnus que ni l'intuition du concepteur ni la subjectivité des testeurs ne représentent. Une proposition infalsifiable ne peut être jugée que par ceux qui l'ont conçue ; une proposition falsifiable peut être jugée par n'importe qui, y compris ceux qui arriveront après.

La seconde : autour d'un même objet se pressent des subjectivités individuelles par milliers, et leur somme devra bien se résoudre quelque part. Il n'existe que deux manières de la résoudre : par la force — l'ancienneté, le veto, le rapport de force, et la partie « La liberté et ses conditions » montrera ce que cela donne — ou par une procédure rationnelle et démocratique. Or la démocratie du design ne porte pas où on la place d'habitude. On ne vote pas sur les solutions : le chapitre « Écouter sans obéir » dira pourquoi l'utilisateur n'a pas autorité sur elles, et le chapitre « Le comité n'inventera rien » ce que produisent les comités. Elle porte sur les *critères* et les *priorités* : ce qui comptera comme « mieux », pour qui, dans quel ordre — c'est-à-dire exactement là où logent les valeurs, et où aucune expertise ne confère d'autorité. Critères délibérés démocratiquement, solutions départagées contre les

critères : chaque étage reçoit le régime qui lui revient. Et l'on tient du même coup ce que la falsifiabilité a de politique : un design infalsifiable n'est discutable que par ses auteurs ; un design falsifiable donne prise à tous ceux qu'il concerne — nul besoin de savoir lire le code pour lire un critère et vérifier qu'il est tenu. La dernière partie de ce livre bâtira sur cette base.

Le rôle de la discipline est donc de préparer le changement d'échelle. Les sciences statistiques savent, dans certains cas, estimer l'incertitude liée au fait d'utiliser des résultats statistiques mesurés sur un petit échantillon d'individus pour inférer le comportement général d'une population plus large. Le raisonnement est mathématique, mais son intuition s'explique sans calcul : puisque l'on ne sait pas dans quelle mesure notre échantillon de test représente fidèlement la population générale (peut-être n'a-t-il pas été choisi complètement au hasard, peut-être est-il le résultat d'un filtrage que personne n'a vu : il est donc biaisé en faveur de l'un ou l'autre des sous-groupes contenus dans la population générale), on ne sait pas non plus dans quelle mesure la connaissance extraite de cet échantillon (nos statistiques) peut être transportée à la population générale pour la décrire fidèlement. Le changement d'échelle amène une erreur non mesurable, mais dont les statistiques peuvent estimer la valeur probable (ou son étendue probable) à un certain niveau de confiance (c'est ce que l'incertitude signifie). La discipline permet non seulement de se préparer à cette erreur inévitable, mais aussi de permettre une rétro-action (une correction) de la population générale lorsque la solution proposée est rendu disponible, à l'aune des objectifs de la solution désirée.

L'objection sérieuse à la falsifiabilité vient de Donald Schön. Observant des architectes, des ingénieurs et des thérapeutes au travail, Schön a montré que le praticien expérimenté mobilise un savoir qu'il ne sait pas énoncer — une « réflexion en cours d'action » qui déborde toute rationalité technique explicitable[6]. L'objection est fondée : une part du savoir de conception est tacite, et quiconque a conçu quoi que ce soit le sait. Mais il faut voir ce que l'objection établit, et ce qu'elle n'établit pas. Que le savoir soit tacite ne le rend pas arbitraire. J'ai une formation de pianiste et j'ai enseigné la musique : l'interprétation est libre, la fausse note ne l'est pas. Le professeur qui corrige un doigté mobilise un savoir largement tacite — il ne peut pas toujours dire *pourquoi* ce doigté est mauvais —, mais sa correction n'est pas une opinion : elle se vérifie dans le tempo tenu, dans la phrase qui passe ou casse. Le tacite est un mode de *possession* du savoir, pas une dispense de *validation*. En conception, c'est la même chose : l'intuition de l'expert propose, les critères disposent. Schön décrit comment les concepteurs savent ; il ne fournit aucune raison de cesser de leur demander des comptes.

Concevoir, c'est une discipline de résolution d'un problème humain, dont on a choisi de faire une lecture technique. Chaque mot y a son importance, et les chapitres qui viennent la déplient : *une discipline* — des critères posés avant la solution, qui donnent aux discussions leur règle d'arrêt (chapitres « Poser le problème » et « Mesurer sans se mentir ») ; *un problème* — sans problème, on ne conçoit pas, on bricole (chapitre « Il n'y a pas de problème technique ») ; *humain* — le problème appartient à quelqu'un, et ce quelqu'un n'est pas la solution (chapitre

« Écouter sans obéir »); *une lecture technique choisie* — la lecture est une décision, pas une fatalité, et d'autres lectures existent. De ces 4 termes, *humain* est celui qui engage le plus loin, et il faudra tout le livre pour aller au bout de ce que cela implique : si le problème appartient à quelqu'un, alors concevoir est ce qu'on doit à ce quelqu'un. Le mot étant fixé, on peut s'occuper de la chose.

---

1. Le français technique maintient la distinction : *conception* pour le processus d'ingénierie, *design* pour sa part graphique. L'anglais recouvre les deux par *design* — ce qui explique bien des malentendus dans les communautés bilingues du logiciel. ↩

2. Simon, H. A. (1969). *The Sciences of the Artificial*. Cambridge, MA : MIT Press (3e éd. 1996) https://mitpress.mit.edu/9780262537537/the-sciences-of-the-artificial/ . « Everyone designs who devises courses of action aimed at changing existing situations into preferred ones. » ↩

3. Rittel, H. W. J. & Webber, M. M. (1973). « Dilemmas in a General Theory of Planning ». *Policy Sciences*, 4(2), 155-169 https://doi.org/10.1007/BF01405730 . Des 10 propriétés qu'ils donnent aux *wicked problems*, la deuxième est ici décisive : « Wicked problems have no stopping rule. » ↩

4. Cross, N. (1982). « Designerly Ways of Knowing ». *Design Studies*, 3(4), 221-227 DOI . ↩

5. Popper, K. R. (1934). *Logik der Forschung*. Vienne : Springer https://search.worldcat.org/title/logik-der-forschung/oclc/263117171 (trad. fr. *La Logique de la découverte scientifique*, Paris : Payot, 1973) — la falsifiabilité comme critère de démarcation entre science et non-science. La transposition à la conception est nôtre : elle emprunte le critère de démarcation, pas la méthode. ↩

6. Schön, D. A. (1983). *The Reflective Practitioner : How Professionals Think in Action*. New York : Basic Books https://books.google.com/books/about/The_Reflective_Practitioner.html?id=E85qAAAAMAAJ . ↩

2. Il n'y a pas de problème technique

On enseigne en école d'ingénieurs que la conception consiste à « apporter une solution technique à un problème technique ». La formule a le mérite de la symétrie et le défaut d'être à moitié fausse. La moitié vraie est violée tous les jours ; la moitié fausse coûte cher à tout le monde, à commencer par l'utilisateur.

La moitié vraie : sans problème, on ne conçoit rien. Il n'y a pas de dissertation sans problématique, et il n'y a pas de conception sans problème — c'est la définition même du chapitre précédent : des situations existantes, des situations préférées, et le chemin de l'une à l'autre. Ôtez le problème, et il ne reste que le chemin, c'est-à-dire l'activité pour elle-même. L'activité pour elle-même porte un nom : la création artistique. Mais l'art produit un objet unique sans usage, tandis que la conception produit un objet utilitaire et répétable, voire un outil de travail dont des gens vont dépendre, qui doit servir plus qu'un seul utilisateur. La conception effectuée avec la démarche de la création artistique se nomme bricolage. Il y a un temps et une place pour le bricolage : lorsqu'on travaille pour soi-même. Il devient un vice de conception quand il se déguise en développement : quand des fonctionnalités que personne n'a demandées s'empilent parce qu'elles amusaient leur auteur, livrées subtilement cassées parce que rien ne définissait ce que « fini » voulait dire. Le folklore des programmeurs a une formule pour cette dérive, attribuée à un certain Louis Srygley dont personne n'a jamais retrouvé la trace :

« sans spécifications ni design, programmer c'est l'art d'ajouter des bugs à un fichier texte vide »[1]. Il y a aussi un argument d'économie, plus prosaïque : concevoir coûte — du temps, de l'argent, de l'attention. Un coût sans problème à résoudre est un gaspillage par définition ; le problème n'est pas une formalité d'entrée, il est ce qui justifie la dépense.

La moitié fausse tient dans l'adjectif : il n'existe pas de problème *technique*. Un problème est un écart entre un état actuel du monde et un état préféré — et la préférence appartient toujours à quelqu'un. Une pile de photographies brutes n'est un problème pour personne ; elle en devient un pour la photographe qui doit livrer 30 images propres avant lundi. Le problème est à elle, pas à la machine : la machine n'a pas d'états préférés. Ce qu'on appelle « problème technique » est un problème humain qu'on a déjà traduit — dont on a choisi de faire une lecture technique — et qui est devenu technique *par cette lecture*, non par sa nature intrinsèque. La nuance a l'air scolastique ; elle est opérationnelle.

La parabole classique le dit en 3 répliques. Un client entre et demande une perceuse. Le vendeur zélé compare les perceuses ; le concepteur demande : pour percer quoi ? — Un trou, pour accrocher un cadre. Le problème n'a jamais été la perceuse, ni même le trou : c'est accrocher le cadre au mur. Et au problème « accrocher un cadre », l'espace des solutions est tout autre — un adhésif double face, un rail à tableaux, et aucun outil à acheter[2]. La leçon n'est pas que le client est sot ; c'est que la demande arrive toujours déjà traduite, et que la traduction du demandeur — il a pensé « perceuse » parce qu'il connaît les perceuses — n'est presque jamais la meilleure. Le premier geste de conception est

de remonter la traduction jusqu'au problème, puis de choisir sa propre lecture.

Or les lectures possibles sont plus nombreuses que l'ingénieur ne l'admet spontanément. Tous les problèmes logiciels ne se résolvent pas en écrivant du code, par exemple. Et toutes les solutions ne sont pas forcément *dans* le logiciel non plus. Quand la même question revient chaque semaine sur les forums d'un logiciel, on peut y répondre par une fonctionnalité — ou par une page de documentation, une vidéo de 3 minutes, une réorganisation de l'interface qui rend la question sans objet. Quand des utilisateurs insuffisamment formés inondent un projet de demandes qui trahissent une incompréhension du domaine, la racine du problème n'est pas dans le code : elle se traite par l'enseignement. Le choix de la lecture est le premier choix de conception, et c'est le plus structurant, parce que tout le reste en découle — y compris la facture.

J'en donne un exemple vécu, parce qu'il montre la mécanique complète. Un assistant architecte avait passé une semaine à se battre avec Visual Basic pour faire afficher à Excel des listes de mobilier extraites d'un logiciel de CAO, pour un nouveau palais de justice à construire. S'agissant de commande publique sur un bâtiment de grande taille, la facture de mobilier devait passer par le processus de validation du budget. Sa lecture du problème : « faire marcher ma macro Excel ». Une heure de questions a suffi à remonter au problème réel — publier des fiches de mobilier lisibles à partir des données du projet — et deux heures à le résoudre par une autre lecture : un script qui convertit l'export CSV du logiciel de CAO en pages HTML, plus 6 heures

d'habillage graphique pour que ça soit joli. Une semaine d'impasse contre une journée de travail : l'écart ne mesure pas une différence de compétence, il mesure le coût d'une mauvaise traduction du besoin. Il se battait avec l'outil qu'il connaissait ; le proverbe dit que quand on tient un marteau, tout finit par ressembler à un clou[3] — et il faut le compléter : celui qui n'a qu'un marteau *traduit* tous ses problèmes en clous, et c'est la traduction, pas le marteau, qui fait les dégâts.

Car la mauvaise lecture a un coût propre, au-delà de l'effort perdu : elle produit du code là où il n'en fallait pas. Un logiciel n'est pas un actif qui s'accumule, c'est un passif qui se maintient — chaque ligne écrite devra être comprise, testée, corrigée et portée par quelqu'un, pendant des années (le chapitre « Le code est un passif » fait les comptes). Répondre par du code à un problème de documentation, c'est convertir un coût ponctuel en dette perpétuelle. L'inverse n'est pas vrai : une page de documentation ne casse pas la compilation, ne fait pas crasher le logiciel, et se corrige en 5 minutes. À lectures équivalentes, celle qui produit le moins de code gagne.

Reste l'objection du puriste technique, qu'il faut prendre au sérieux parce qu'elle structure encore la formation des ingénieurs : l'ingénierie s'occuperait des machines, et les humains s'adapteraient — ils ont toujours fini par s'adapter. L'objection a pour elle une part d'histoire : oui, les humains s'adaptent, c'est même leur talent principal. Mais elle échoue 3 fois. D'abord chez les siens : c'est du cœur du design industriel qu'est venue la réfutation la plus nette, celle de Don Norman — ce qu'on appelle « erreur humaine » est presque toujours une erreur de

conception, et blâmer l'utilisateur est le moyen le plus sûr de ne jamais corriger le système[4]. Ensuite par les coûts : l'adaptation n'est pas gratuite ; elle est payée par quelqu'un, en temps d'apprentissage, en erreurs, en fatigue — un coût réel, déporté hors du budget du projet vers des gens qui n'ont pas signé pour le porter. Enfin par incohérence : même les critères que le puriste tient pour purement techniques — la performance, la robustesse — n'ont de sens que rapportés à des fins humaines. Une performance, c'est du temps rendu à quelqu'un ; une robustesse, c'est une confiance que quelqu'un peut accorder à l'outil. Débranchez l'humain, et il ne reste rien à optimiser. Et cet humain-là a une propriété que ce livre ne cessera de retrouver : il n'a pas choisi la machine, elle lui est tombée dessus, et elle l'a éloigné du matériau qu'il manipule, en transformant le « coup de main » permettant un ajustement contextuel en procédures figées et standardisées.

Le problème est humain, sa traduction est un choix, et ce choix se refait, se défait, se questionne. Mais si le problème appartient à quelqu'un, il faut bien aller le chercher chez lui — et c'est là que les ennuis commencent, car celui qui vit le problème est aussi le plus mal placé pour le formuler. Écouter sans obéir : c'est l'objet du chapitre suivant.

---

1. « Without requirements or design, programming is the art of adding bugs to an empty text file. » Citation omniprésente dans le folklore professionnel, systématiquement attribuée à Louis Srygley — dont aucune source primaire ni aucune

trace biographique n'a pu être établie. Elle est citée ici pour ce qu'elle est : de la sagesse anonyme de métier. ↩

2. La parabole descend d'un slogan publicitaire américain des années 1940 : « L'an dernier, un million de perceuses d'un quart de pouce ont été vendues — non que les gens voulussent des perceuses, mais parce qu'ils voulaient des trous d'un quart de pouce. » Attribuée à Leo McGivena (1947), popularisée par Theodore Levitt qui la citait en créditant McGivena (*The Marketing Mode*, 1969) https://openlibrary.org/books/OL4433438M/The_marketing_mode, puis massivement mal attribuée à Levitt lui-même. Clayton Christensen en a tiré la théorie moderne des « jobs to be done » : on n'achète pas un produit, on l'embauche pour faire un travail (« Marketing Malpractice », *Harvard Business Review*, 2005) https://hbr.org/2005/12/marketing-malpractice-the-cause-and-the-cure. ↩

3. « Loi de l'instrument », formulée successivement par Abraham Kaplan (*The Conduct of Inquiry*, 1964) https://doi.org/10.4324/9781315131467 et Abraham Maslow (*The Psychology of Science*, 1966) https://openlibrary.org/books/OL41390694M/The_psychology_of_science : « il est tentant, quand on ne possède qu'un marteau, de traiter toute chose comme un clou. » ↩

4. Norman, D. A. (2013). *The Design of Everyday Things*, éd. revue et augmentée. New York : Basic Books, chap. 2 https://www.hachettebookgroup.com/titles/don-norman/the-design-of-everyday-things/9780465050659/. Norman y décrit aussi l'auto-accusation des utilisateurs face aux mauvaises

interfaces — ils se blâment eux-mêmes pour des fautes de conception — et forge pour ce mécanisme le terme de *taught helplessness*, l'impuissance enseignée. Nous y reviendrons longuement au chapitre « Ce que l'entre-soi produit ». ↩

3. Écouter sans obéir

Le chapitre précédent laissait le concepteur devant une difficulté : le problème appartient à quelqu'un, il faut donc aller le chercher chez lui — mais celui qui vit le problème est le plus mal placé pour le formuler. Deux erreurs symétriques attendent ici. La première est d'obéir : faire ce que l'utilisateur demande, comme il le demande. La seconde est de ne plus écouter du tout : décider à sa place, du haut de l'expertise. La première produit des logiciels difformes ; la seconde, des logiciels pour personne. La voie médiane tient en 3 mots : écouter sans obéir.

Entendons-nous sur ce que l'utilisateur *peut* dire. Jamais un utilisateur ne vient exposer son besoin ; il vient exposer ce qu'il veut, et les deux ne coïncident presque jamais — il arrive même que le besoin soit le contraire de la demande. Ce n'est pas une infirmité, c'est une conséquence de sa position : pour imaginer une solution, il faut connaître l'espace des solutions, et l'utilisateur n'en connaît qu'un point — l'outil qu'il a entre les mains, qu'il va être tenté de modifier marginalement pour tous ses besoins plutôt que d'en choisir un autre, plus adapté à la nouvelle tâche qu'il doit accomplir (voir le chapitre précédent). On prête à Henry Ford le mot fameux — « si j'avais demandé aux gens ce qu'ils voulaient, ils m'auraient répondu : des chevaux plus rapides » ; il ne l'a très probablement jamais dit, la première attribution connue datant d'un demi-siècle après sa mort[1], mais l'apocryphe a prospéré parce qu'il condense une observation

exacte : la demande est toujours formulée dans le vocabulaire des solutions connues, et sera presque toujours une variante mineure de l'existant (la fonctionnalité du concurrent, le bouton déplacé, le cheval plus rapide). Exécuter ces demandes une à une, c'est rafistoler l'existant jusqu'à la créature de Frankenstein — un logiciel fait de greffes, dont aucune ne répond au problème parce que personne ne l'a posé. Une vision romanticisée de la chose pourrait y voir un outil « qui transcende sa fonction première », qui échappe aux objectifs de ses concepteurs quand ses utilisateurs se l'approprient. Cela peut être vrai dans une certaine mesure, mais il faut se demander si ajouter des fonctionnalités d'ouvre-boîte à un tournevis n'est pas juste en train d'en faire un tournevis plus dangereux à utiliser, moins pratique à tenir, moins solide pour appliquer un couple de serrage spécifié et donc, finalement, plus mauvais à sa tâche initiale. Si l'on n'a pas arrêté de vendre des tournevis, des clés hexagonales, et toute la panoplie après avoir inventé le couteau suisse ou le multi-outil Leatherman, c'est bien que ces outils polymorphes sont pratiques pour dépanner ponctuellement, mais inefficaces pour une utilisation intensive et exigeante.

La psychologie de la qualité a caractérisé pourquoi l'utilisateur a tant de mal à définir lui-même son besoin. Noriaki Kano distinguait 3 classes d'attentes[2] : les attentes *implicites*, si évidentes pour l'utilisateur qu'il ne pensera jamais à les énoncer — leur absence rend furieux, leur présence passe inaperçue ; les attentes *exprimées*, sur lesquelles il sait se prononcer ; et les attentes *latentes*, qu'il ne peut pas formuler parce qu'il n'imagine pas qu'elles soient possibles. Interroger directement ne

moissonne que la bande du milieu. Le socle et le sommet — ce qui va sans dire et ce qui dépasse l'imagination — sont invisibles au questionnaire, et ce sont précisément eux qui font les outils sûrs et les outils remarquables.

À cette première limite s'en ajoute une seconde, plus insidieuse : quand la discussion quitte l'utilisateur réel pour « l'utilisateur moyen », elle quitte le réel tout court. L'armée de l'air américaine l'a appris en 1952 : sur 4 063 aviateurs mesurés, le nombre de ceux qui étaient proches de la moyenne sur les 10 dimensions du cockpit était exactement zéro[3]. L'homme moyen est une fiction statistique ; concevoir pour lui, c'est concevoir pour personne — la solution des cockpits fut le réglable, pas le moyen. L'utilisateur moyen des discussions de projet est pire encore que cette fiction : n'étant mesuré par rien, il devient élastique — il prend la forme de l'argument qu'il doit servir, et chacun invoque le sien[4]. La parade est connue : des cas réels, nommés, situés — cette photographe, ce volume, cette échéance — et, quand il faut généraliser, des archétypes construits à partir de cas réels, jamais des moyennes.

Un garde-fou avant de quitter le cockpit, car la leçon se lit trop vite. « Réglable » n'est pas un mot magique : chaque réglage est un degré de liberté, chaque degré de liberté une pièce mobile, et chaque pièce mobile une source de panne, une source d'usure, un coût de fabrication et de maintenance, et un cas de test de plus — en logiciel, chaque option multiplie les combinaisons à vérifier, et le chapitre « L'éloge du simple » fera le procès de cette inflation-là. Le siège du cockpit ne réglait pas tout ; il réglait les dimensions où les corps varient de façon *mesurée*, et où cette variation

empêche la tâche — atteindre les commandes. C'est le critère complet, et il a deux faces : un réglage se paie, et il ne s'achète que là où la variation entre utilisateurs réels est démontrée et bloque le travail ; partout ailleurs, la bonne réponse reste une valeur fixe, bien choisie une fois. L'adaptabilité est un budget, pas une vertu, et ce chapitre ne plaide pas pour plus de boutons.

Que reste-t-il alors à écouter ? Beaucoup — mais pas les déclarations. La première règle de l'utilisabilité, selon Jakob Nielsen : ne pas écouter ce que les utilisateurs *disent*, regarder ce qu'ils *font*[5]. Les déclarations sont reconstruites, complaisantes, prédictives — et les prédictions que les gens font sur leur propre comportement ne sont que faiblement corrélées à leurs actes : la préférence déclarée ne prédit qu'un quart de la performance mesurée. Le geste d'écoute sérieux a été codifié par l'enquête contextuelle : aller sur le lieu du travail réel, et s'y mettre en position d'*apprenti* — l'utilisateur est le maître, non du logiciel, mais de son métier, et on apprend son métier comme un apprenti apprend : en regardant faire, en demandant pourquoi, sur des cas concrets et présents[6]. Et lorsqu'il faut malgré tout interroger, la règle d'Ulwick borne la question : demander les *résultats* attendus — qu'est-ce qui, à la fin, doit être vrai ? — jamais les solutions ; les produits bâtis sur les solutions spécifiées par les clients échouent avec une régularité remarquable[7].

Tout ceci se condense en un critère opérationnel, qui servira dans tout le livre : **l'utilisateur a autorité sur ses buts, son contexte et ses douleurs — jamais sur les solutions.** Est recevable son témoignage : ce qu'il fait, comment il le fait, ce que ça lui coûte — du concret, au présent ou au passé. Est irrecevable

sa prescription : ce qu'il faudrait faire, ce qu'il utiliserait sûrement, ce que veulent « les gens » — du normatif, au futur, au conditionnel. Reste l'opinion — le jugement porté sur une solution proposée ou livrée. Avant de l'avoir essayée, cette opinion n'est qu'une prédiction de plus, et tombe avec les autres ; après un essai réel et assez long pour dépasser la gêne du changement d'habitude, elle est devenue un témoignage : elle rentre dans la première catégorie, et compte.

Deux situations vécues montrent le critère au travail. Un mainteneur de documentation — qui connaissait le logiciel mieux que moi, des années de manuel derrière lui — reprochait à mon interface de demander, pour certaines opérations, plus de clics que l'ancienne, qui réalisait de nombreuses tâches dangereuses sur simple survol de l'élément à éditer. Témoignage recevable : la douleur est réelle, comptée, située. Prescription irrecevable : « remettez comme avant ». Car derrière le compte de clics se cachaient d'autres besoins, muets au sens de Kano — ne pas détruire de données sans prévenir, juste par un survol malheureux, rester transparent sur la cible des opérations de lecture/écriture pour celui qui arrive — et l'arbitrage entre ces besoins est un travail de conception, pas un vote (le chapitre « Mesurer sans se mentir » fera les comptes). De même le jour où j'ai regroupé en un seul endroit les commandes de masquage — les masques sont les zones qui restreignent un réglage à une partie de l'image — au lieu de les répéter dans chacun des modules qui s'en servent. La protestation fut générale, et son témoignage exact : l'outil est plus loin qu'avant, demande plus de clics, et il faut traverser l'interface pour l'atteindre. Mais obéir —

remettre les commandes dans chaque module — aurait allongé chaque module, sur des écrans dont la télémétrie montrait qu'un utilisateur sur 3 manque déjà de hauteur d'écran pour en afficher un seul en entier. Le témoignage disait le coût du changement ; il ne disait rien du coût de l'alternative, que seul le concepteur voyait. Écouter la douleur, refuser l'ordonnance : les deux gestes, ensemble, à chaque fois.

Reste l'objection la plus sérieuse, et elle est double. L'école scandinave du *design participatif* soutient que les utilisateurs doivent co-concevoir, pour deux raisons de nature différente[8]. Une raison politique : la technologie restructure le travail, et ceux dont le travail est restructuré ont un droit démocratique à peser sur elle — ce n'est pas un argument d'efficacité, et aucune étude d'utilisabilité ne peut le réfuter. Une raison épistémique : le savoir-faire des métiers est tacite, il échappe aux entretiens comme à l'observation, et seuls des dispositifs de *conception par le faire* — maquettes, prototypes à manipuler — le font parler. L'objection épistémique, à l'examen, confirme le critère plus qu'elle ne le renverse : le design participatif ne fait jamais *spécifier verbalement* une solution par l'utilisateur — il change le support pour que la compétence tacite s'exprime, c'est-à-dire qu'il enrichit le témoignage ; le maquettage est une enquête contextuelle par d'autres moyens. Quant à l'objection politique, elle est fondée, et ce livre y répondra sur son terrain : la dernière partie défend la co-propriété des *buts* et des budgets par ceux qui utilisent l'outil — décider ensemble *ce qui* mérite d'être résolu est un droit ; le *comment* reste un métier. Un mot enfin des utilisateurs-pionniers, dont on sait qu'ils innovent réellement

dans certains domaines : ils ne contredisent pas le critère, ils en changent le sujet — celui qui conçoit son propre outil est devenu concepteur, avec les devoirs du poste.

Une division du travail a pourtant une clause de réciprocité, qu'il faut énoncer avant de refermer ce chapitre, car son oubli renverse tout. Si le concepteur doit écouter l'utilisateur sans lui obéir, encore faut-il que son propre témoignage — sur ce qu'une solution coûte, sur ce qu'un changement casse, sur ce qui est faisable — soit à son tour entendu par le collectif. Faute de quoi la formule s'inverse en *obéir sans être entendu.* Le concepteur n'est plus l'autorité sur l'espace des solutions ; il devient l'exécutant d'un espace que d'autres décident, sommé d'en réparer les dégâts sans droit de parole sur les causes. La forme pure de ce renversement m'a été adressée un soir d'août 2022, quand une refonte du système d'entrées avait de nouveau brisé un composant que j'avais réparé 3 ans plus tôt : à mes objections, un contributeur répondit, en 3 messages — trouve le bug, répare-le, arrête de te plaindre[9]. La phrase, sous les deux formes, est un objet théorique parfait. Elle dénie au concepteur le seul registre que ce chapitre lui accorde sans réserve — le témoignage sur le réel technique — tout en lui imposant la seule chose que le chapitre attend de lui : l'exécution. C'est « écouter sans obéir » retourné comme un gant contre celui qui fait le travail. L'étude des organisations connaît ce régime sous le nom de *silence organisationnel* : un climat où le désaccord remonté vers le haut est systématiquement découragé, jusqu'à ce que ceux qui savent cessent de parler et partent[10]. Le mécanisme complet — ce qui arrive à celui dont la parole interne est étouffée, et pourquoi il

finit par partir plutôt que par se taire — relève de la dernière partie de ce livre ; il suffit ici de retenir que la division du travail proposée n'est pas une hiérarchie déguisée, mais un contrat à deux obligations mutuelles, et que n'importe laquelle de ces obligations non remplie dégénère la conception en domination, indépendamment de qui l'exerce et de qui la subit. Et il faut noter la portée de cet énoncé, qui déborde de loin le cas d'espèce : toute relation où l'un peut et l'autre a besoin bascule en domination dès qu'elle cesse de rendre des comptes — c'est vrai du service rendu à l'utilisateur autant que du travail exigé de celui qui produit.

Écouter sans obéir n'est donc pas une posture de défiance, c'est une division du travail. À l'utilisateur le problème, sa chair, son contexte ; au concepteur la traduction et l'espace des solutions ; à l'usage honnête le verdict — et à chacun le droit d'être entendu dans son registre. Encore faut-il ne pas traiter la plainte là où elle se montre. Ce que l'utilisateur signale est un symptôme, pas une cause, et la première tâche est d'y remonter avant d'agir. C'est l'objet du chapitre suivant.

---

1. Aucune trace du mot chez Ford ni chez ses contemporains ; première attribution connue en 1999, 52 ans après sa mort (Quote Investigator, « Faster Horse », 2011 https://quoteinvestigator.com/2011/07/28/ford-faster-horse/ ; Vlaskovits, P., « Henry Ford, Innovation, and That "Faster Horse" Quote », *Harvard Business Review*, 2011 https://hbr.org/2011/08/henry-ford-never-said-the-fast). ↩

2. Kano, N., Seraku, N., Takahashi, F. & Tsuji, S. (1984). « Attractive Quality and Must-Be Quality ». *Hinshitsu — Journal of the Japanese Society for Quality Control*, 14(2), 39-48. https://doi.org/10.20684/quality.14.2_147 Les 3 classes originales : *must-be*, *one-dimensional*, *attractive*. ↩

3. Daniels, G. S. (1952). *The "Average Man" ?* Technical Note WCRD 53-7, Aero Medical Laboratory, Wright-Patterson Air Force Base. https://apps.dtic.mil/sti/citations/tr/AD0010203 Sur 4 063 sujets, aucun ne se situait dans le tiers médian des 10 dimensions à la fois. Voir aussi Rose, T. (2016). *The End of Average*. New York : HarperOne. https://search.worldcat.org/title/926105728 ↩

4. L'« utilisateur élastique » est d'Alan Cooper : *The Inmates Are Running the Asylum*. Indianapolis : Sams, 1999 https://search.worldcat.org/title/41237183 — avec sa parade, les *personas* : des archétypes précis construits sur des utilisateurs réels. ↩

5. Nielsen, J. (2001). « First Rule of Usability ? Don't Listen to Users ». Nielsen Norman Group. https://www.nngroup.com/articles/first-rule-of-usability-dont-listen-to-users/ La corrélation préférence déclarée / performance mesurée, d'environ 0,44, provient de Nielsen, J. & Levy, J. (1994). « Measuring Usability : Preference vs. Performance ». *Communications of the ACM*, 37(4). https://doi.org/10.1145/175276.175282 ↩

6. Beyer, H. & Holtzblatt, K. (1997). *Contextual Design : Defining Customer-Centered Systems*. San Francisco : Morgan Kaufmann. https://search.worldcat.org/title/37443490 Le modèle d'entretien y est explicitement celui du maître et de l'apprenti. ↩

7. Ulwick, A. W. (2002). « Turn Customer Input into Innovation ». *Harvard Business Review*, janvier 2002. https://hbr.org/2002/01/turn-customer-input-into-innovation ↩

8. Ehn, P. (1988). *Work-Oriented Design of Computer Artifacts*. Stockholm : Arbetslivscentrum https://books.google.com/books/about/Work_oriented_Design_of_Computer_Artifac.html?id=7dJQAAAAMAAJ ; Sundblad, Y. (2011). « UTOPIA : Participatory Design from Scandinavia to the World », in *History of Nordic Computing 3*. Berlin : Springer. https://doi.org/10.1007/978-3-642-23315-9_20 Sur la filiation avec l'innovation par les utilisateurs-pionniers : Björgvinsson, E., Ehn, P. & Hillgren, P.-A. (2010). « Participatory Design and "Democratizing Innovation" ». *Proceedings of the 11th Biennial Participatory Design Conference*, ACM. https://doi.org/10.1145/1900441.1900448 ↩

9. Journal du salon Matrix public du projet (pont du canal IRC), export du 23 juillet 2026 en ma possession. Le 8 août 2022 à 22 h 06, en 3 messages : « find the bug », « fix it », « stop fucking bitching ». La séquence que la pièce établit est celle qui importe ici : un composant que j'avais réparé en 2019 ; cassé par une refonte que je n'avais pas pu relire ; à ma

charge de nouveau — trouve, répare, et tais-toi — sans qu'aucune aide ne soit proposée. Le contexte complet de la soirée est versé au dossier du chapitre « Le cercle fermé ». ↩

10. Morrison, E. W. & Milliken, F. J. (2000). « Organizational Silence : A Barrier to Change and Development in a Pluralistic World ». *Academy of Management Review*, 25(4), 706-725. https://doi.org/10.2307/259200 Voir aussi Van Dyne, L., Ang, S. & Botero, I. C. (2003). « Conceptualizing Employee Silence and Employee Voice as Multidimensional Constructs ». *Journal of Management Studies*, 40(6), 1359-1392. https://doi.org/10.1111/1467-6486.00384 ↩

Partie II — La discipline

4. La racine et le symptôme

Un problème se montre où il fait mal, pas où il est. Le médecin le sait — la douleur projetée irradie loin de sa lésion —, l'ingénieur l'oublie : l'endroit où le défaut se manifeste est un indice, pas une adresse. Le chapitre précédent apprenait à recueillir les douleurs ; celui-ci apprend à ne pas les traiter sur place. C'est un art délicat, parce que tout — l'urgence, l'économie, la psychologie — pousse à agir là où l'on voit et à ignorer le reste.

L'enjeu se chiffre. Un problème racine porte des branches : traité à la racine, il les résout toutes d'un coup, et la solution est généralement plus simple et plus générale que la somme des rustines qu'auraient exigées ses manifestations. Traité aux branches, il repousse — et chaque rustine ajoute de la surface au système : du code de plus à comprendre, des interactions de plus à casser. La forme extrtême de cette dérive est connue de tout mainteneur : le jeu de la taupe (*whack-a-mole*), où chaque correction fait surgir un défaut ailleurs, jusqu'au point où le logiciel ne *contient* plus de bugs — il en *est* un. À ce stade, aucun traitement local n'a plus de sens ; c'est le signe que les symptômes ont été soignés assez longtemps pour que la maladie ait tout envahi. Le traitement symptomatique a parfois une élégance qui le trahit : voir un logiciel « régler » la lenteur de son

démarrage en affichant un écran d'accueil pendant qu'il charge, c'est voir un symptôme habillé en fonctionnalité — le temps de chargement n'a pas diminué d'une milliseconde, on a seulement peint la salle d'attente.

Si le traitement symptomatique est si mauvais, pourquoi est-il si universel ? Parce qu'il est rationnel — localement, à court terme, pour chacun. Le philosophe Abraham Kaplan appelait cela le principe de la recherche de l'ivrogne : l'homme qui cherche ses clés sous le réverbère, non qu'il les ait perdues là, mais « parce que c'est là qu'il y a de la lumière »[1]. Ceci est connu sous le nom d'effet lampadaire ou de biais du lampadaire. On agit où l'on voit, parce que voir rend l'action possible et bon marché, et parce qu'agir nous donne un sentiment de contrôle sur la situation. On remarque que s'asseoir sur ses mains en attendant d'avoir suffisamment bien compris le problème pour être raisonnablement sûr qu'on tienne bien sa cause est rarement une option audible, quand elle ne passe pas carrément pour un sophisme de la solution parfaite. Or il ne s'agit pas ici d'opposer une solution symptomatique, temporairement et localement meilleure, à l'inaction pure et simple, au motif que la solution symptomatique ne serait pas parfaite : il s'agit de ne rien faire qui puisse endommager à long terme le système global tant qu'on n'a pas fini de cartographier l'étendue du problème et son origine, sans quoi les solutions symptomatiques vont devoir se multiplier à différents étages du système.
La dynamique des systèmes a formalisé la suite : dans l'archétype que Peter Senge nomme le *report de charge*, la solution symptomatique soulage plus vite et moins cher que la solution

fondamentale — et chaque soulagement réduit la pression d'entreprendre la solution fondamentale, dont la capacité *s'atrophie* ; l'organisation devient dépendante de sa rustine comme d'un analgésique, à doses croissantes[2]. Personne n'a été stupide à aucun moment : chacun a suivi la pente du résultat maximal (ou, du moins, maximalement visible) pour un effort minimal, le tout vu à court terme. C'est précisément pour cela qu'il faut une discipline — on ne remonte pas une pente par inclination.

La discipline existe, et elle vient des usines. Chez Toyota, Taiichi Ohno l'avait réduite à une consigne d'une ligne : devant un problème, demander *pourquoi* 5 fois. Une machine s'arrête. Pourquoi ? — Le fusible a sauté, surcharge. Pourquoi ? — Le palier était mal lubrifié. Pourquoi ? — La pompe de lubrification débitait mal. Pourquoi ? — Son axe était usé. Pourquoi ? — Faute de crépine, des copeaux de métal entraient dans la pompe. Remplacer le fusible remet la machine en marche aujourd'hui et garantit la même panne dans 3 mois ; poser une crépine la fait disparaître pour toujours[3]. Le fusible est l'endroit où le problème se voit ; la crépine, l'endroit où il se décide. Entre les deux, 5 questions — et l'art est moins dans le comptage que dans le franchissement : chaque *pourquoi* sérieux change d'étage, du composant au sous-système, du sous-système à l'architecture, de l'architecture au processus qui a produit l'architecture. Le même mouvement vaut pour les besoins, qui sont des poupées russes : si deux besoins semblent encore se contredire, c'est en général qu'on tient des ramifications, pas la racine — on n'a pas remonté assez haut.

La sécurité industrielle a donné à cette verticalité ses noms définitifs. James Reason distingue les défaillances *actives* — l'erreur du bout de chaîne, visible, immédiate, imputable à quelqu'un — des conditions *latentes* : des « pathogènes résidents » installés dans le système longtemps avant, par des décisions de conception, d'organisation, de budget, et qui attendent leur occasion[4]. L'accident n'est presque jamais l'erreur active seule ; il est la rencontre de l'erreur active avec des conditions latentes que personne ne regardait, parce qu'elles ne faisaient mal nulle part. Où l'on voit le problème : au bout de chaîne. D'où il vient : de l'amont, des années plus tôt.

J'ai été, pendant quelques années, le bout de chaîne. Le module de courbe de rendu que j'avais écrit — *filmic*, qui comprime la dynamique de la scène dans celle de l'écran — se plaçait, par construction, en fin de pipeline : il recevait ce que tous les modules précédents avaient produit, et le rendait visible. Il éclaircissait les ombres, donc il révélait le bruit du capteur que les ombres bouchées cachaient ; il préservait la saturation, donc il rendait tels quels les hautes lumières écrêtées que l'amont avait laissé virer au magenta, là où les anciens rendus les délavaient pudiquement ; et il héritait, à la manière du *garbage in, garbage out*, de chaque défaut de ce qui le précédait. Les plaintes m'arrivèrent donc toutes — bruit, magenta, couleurs fausses —, adressées au module qui les montrait, jamais à ceux qui les fabriquaient : un fil de discussion demandait pourquoi filmic « rend si bizarrement les zones écrêtées »[5], alors que l'écrêtage est un fait du capteur, traité — ou non — bien en amont. C'est reprocher au thermomètre la fièvre. Le mécanisme est décrit : le

jugement causal ordinaire désigne la condition *anormale* — ce qui vient de changer, ce qu'on peut le plus facilement imaginer absent — et laisse hors champ les conditions de fond qui ont fait le travail[6] ; dans une chaîne, il crédite le dernier maillon qui a fait la différence visible[7] ; et l'on tient rigueur, même à un porteur de nouvelles dont on sait qu'il n'y est pour rien, de la nouvelle qu'il apporte[8]. La sécurité industrielle a son mot pour cela : blâmer le *bout pointu* — l'opérateur, le composant visible — pour des conditions latentes fabriquées au *bout épais*[9]. Et le débogage a mesuré la distance : dans le recensement de leurs pires bugs, des programmeurs professionnels imputent plus de la moitié de la difficulté au *gouffre* temporel et spatial entre la cause et le symptôme[10]. Le module en fin de chaîne recevra donc les reproches de tout le pipeline ; il faut s'y attendre, et ne pas y répondre en lui faisant cacher — par exemple en délavant les hautes lumières — ce que l'amont a cassé : le défaut resterait, on ne le verrait plus.

J'ai vécu la version logicielle de ce théorème. Un cache de calcul — la pièce qui évite de relancer des calculs lourds que l'exécution précédente avait déjà faits — traînait des bugs vieux de 10 ans : des images fantômes, des recalculs intempestifs, des incohérences qu'on corrigeait une à une à mesure qu'elles se montraient. L'une de ces rustines — un rafraîchissement forcé de tout l'affichage par vidange intégrale du cache, ce qui annulait toute sa raison d'être — a été ajoutée, retirée pour cause de consommation, réintroduite derrière une temporisation arbitraire, blindée après des plantages, puis finalement supprimée : 10 ans de traitement symptomatique d'un unique symptôme. La remontée, quand elle

a enfin eu lieu, a suivi les étages de Ohno : réparer le cache exigeait de réparer le pipeline de calcul ; réparer le pipeline exigeait de rendre visible le cycle de vie des données — qui est valide, qui est périmé, qui décide. Les bugs du cache n'étaient pas des bugs *dans* le cache : ils étaient les symptômes d'un moteur dont les responsabilités étaient mélangées. Une fois le cycle de vie explicité, les corrections ont cessé d'être des rustines — chacune *retirait* du code au lieu d'en ajouter. J'en ai tiré une règle pratique, plus efficace que toutes les exhortations à « chercher la cause racine » : **refuser toute correction qui ne simplifie pas le code.** Une rustine ajoute — un cas particulier, un verrou, une temporisation ; un traitement de racine retranche, parce qu'il supprime le malentendu qui rendait ces défenses nécessaires. Devant une correction proposée, il suffit donc de regarder ce qu'elle fait au code : si elle en ajoute, on n'est probablement pas encore descendu à la racine. Et si elle ajoute du code, il convient encore de se demander si c'est pour complexifier l'existant et ses heuristiques, ou au contraire pour ajouter des outils de diagnostic, des systèmes de secours et de détection d'exception *en amont*. Le nombre de lignes de code est un indicateur parmi d'autres, pas la métrique à optimiser. La métrique à optimiser, ici, est la complexité terminale du système, qui se règle en gérant une fois les anomalies, le plus tôt possible dans les entrées du programme, de sorte à éviter de blinder toutes les sorties possibles avec des solutions de repli pour tous les cas d'erreurs ou d'exceptions que l'amont a laissés passer.

La chasse à la racine a cependant un excès inverse, qu'il faut prévenir : chercher une cause à ce qui n'en a pas. Tout système

fluctue — une chaîne de production ne sort jamais deux pièces rigoureusement identiques, un logiciel n'exécute jamais deux fois la même tâche exactement dans le même temps. W. Edwards Deming appelait *causes communes* cette variation-là : le bruit de fond du système, qui n'est imputable à aucun événement particulier ; et il réservait le nom de *causes spéciales* aux écarts derrière lesquels il y a réellement un événement identifiable — la panne de Ohno en est une. La distinction commande la conduite à tenir, car corriger un à un des écarts qui ne sont que du bruit ne stabilise rien : cela empire les choses. Deming le démontrait avec l'expérience de l'entonnoir : on laisse tomber des billes à travers un entonnoir au-dessus d'une cible, et l'on « corrige » la position de l'entonnoir après chaque bille tombée, selon l'écart entre son point de chute et la cible, en espérant ainsi corriger la trajectoire par raffinements successifs. Toutes les règles de correction essayées dispersent les impacts davantage que de ne toucher à rien : en moyenne, le système non corrigé était déjà à son optimum de stabilité, la correction après chaque événement individuel ne faisait que le rendre plus instable[11]. On parle alors de surcorrection.

Poursuivre la cause d'une fluctuation normale n'est pas de la rigueur, c'est du dérèglement. La leçon ne contredit pas les 5 *pourquoi* : elle en fixe la condition d'emploi. Avant de remonter d'un symptôme vers sa cause, s'assurer que le symptôme *a* une cause — que l'écart sort du bruit normal du système. S'il n'en sort pas, il n'a pas de racine propre ; seule la variation d'ensemble en est une, et elle ne se réduit qu'en reconcevant le système — ou pas du tout. Distinguer l'écart significatif du bruit est un

problème de mesure et de statistiques ; les chapitres « Mesurer sans se mentir » et « Compter les absents » donnent les instruments.

Le bruit n'est pas la seule fausse cause ; la coïncidence en est une autre, et elle est plus persuasive parce qu'elle vient avec une histoire. La logique la nomme depuis l'Antiquité — *post hoc ergo propter hoc* : après cela, donc à cause de cela — et le logiciel lui offre un terrain idéal, parce que tout changement y est daté, public et suivi. Publiez une modification : tout ce qui va mal dans les jours suivants lui sera imputé, y compris dans des parties du programme qu'elle ne touche pas et ne peut pas toucher. Je m'attendais à ce que la formation protège de ce réflexe ; l'expérience m'a montré l'inverse. Ceux qui m'ont servi ce raisonnement avec le plus d'aplomb étaient des scientifiques et des programmeurs expérimentés, âgés de cinquante à soixante ans, qui suivaient les changements du logiciel de bien plus près que la plupart des utilisateurs. Ils n'apportaient pas un symptôme : ils apportaient un diagnostic, avec l'explication qui va avec, et le crédit nécessaire pour qu'on l'écoute. Et on l'écoute, précisément parce qu'il vient de gens compétents. On remonte leur piste pendant une demi-journée, parfois davantage, sans rien trouver ; ce n'est qu'à ce moment-là qu'on commence à douter du diagnostic lui-même, pour finir par constater qu'il était faux depuis le début — mais il était plausible, et il venait d'une personne d'expérience. La psychologie expérimentale a documenté chaque étage de cette mécanique. Pour attribuer une cause, nous ne comptons pas les cas où l'effet suit la cause présumée et ceux où il ne la suit pas ; nous nous fions à des

indices — ce qui est venu juste avant, ce qui est proche, ce qui se remarque. En laboratoire, les sujets désignent comme cause l'événement qui a précédé l'effet de peu, même quand les fréquences observées montrent qu'un autre événement en est responsable ; et plus on éloigne dans le temps la cause réelle de son effet, moins ils croient qu'elle en est la cause[12]. Les compétences n'immunisent pas : des psychologues mathématiciens réunis en congrès tenaient les mêmes intuitions fausses que le public sur les petits échantillons ; la plupart des biais de raisonnement sont indépendants de l'aptitude cognitive ; une numératie supérieure aggrave parfois la lecture motivée des données ; et l'intuition d'expert n'est fiable que dans les domaines qui la corrigent par un retour rapide et régulier — ce qu'un logiciel qu'on ne débogue pas soi-même n'offre pas[13]. Deux choses, en revanche, font leur effet, et elles s'additionnent. L'assurance : entre deux avis, on suit celui qui est donné avec le plus de conviction, pas celui qui s'est le plus souvent avéré juste. Et le titre : le même message convainc davantage quand il vient d'une source jugée compétente, quel que soit son contenu[14]. Une erreur affirmée avec aplomb par un expert cumule les deux primes, et va plus loin que la même erreur avancée par n'importe qui d'autre. Le dernier étage est celui qui coûte : une hypothèse plausible, une fois posée, oriente toute l'enquête — les cliniciens à qui l'on souffle un diagnostic *perçoivent* d'autres signes, l'étiquette prend un *élan diagnostique* que les suivants ne délogent plus, et l'arrêt prématuré de l'enquête — s'arrêter à la première explication satisfaisante — est la première cause cognitive d'erreur médicale[15] ; les testeurs de logiciels écrivent surtout des tests qui confirment, et le biais de confirmation

mesuré chez les développeurs prédit la densité de défauts de leur code[16]. La conséquence pratique rejoint le critère du chapitre « Écouter sans obéir » : le rapporteur a autorité sur le symptôme — ce qu'il a vu, quand, comment le reproduire —, jamais sur la cause. Un diagnostic non vérifié n'est pas une information de plus, c'est une ancre ; ceux qui font métier de résoudre des bugs le savent, puisqu'ils classent les étapes de reproduction et les traces d'exécution en tête de ce qu'ils attendent d'un rapport, et la théorie du rapporteur nulle part[17]. La meilleure contribution d'un expert à un problème qu'il n'a pas instrumenté lui-même est donc la plus modeste : décrire, dater, reproduire — et garder l'hypothèse pour lui tant qu'il ne l'a pas testée.

Au sommet de la remontée, il y a mieux que résoudre. Russell Ackoff distinguait 4 traitements d'un problème :

- l'*absoudre* (l'ignorer en espérant qu'il passe),
- le *résoudre* (trouver un résultat acceptable),
- le *solutionner* (trouver l'optimum),
- le *dissoudre* — reconcevoir le système ou son environnement de sorte que le problème ne puisse plus s'y produire[18].

La crépine de Ohno ne résout pas la panne : elle la dissout. Le cycle de vie explicité ne corrige pas les bugs du cache : il en supprime la catégorie. La dissolution est l'horizon du concepteur : non pas gagner contre le problème, mais lui retirer son habitat.

Toute une école de la sécurité des systèmes complexes objecte à cela que la « cause racine » est une erreur de catégorie. Sidney Dekker : la cause racine n'est pas une propriété du monde, c'est

l'endroit où l'enquête *a décidé de s'arrêter* — un artefact du budget et de la politique de l'investigation. Nancy Leveson : le modèle de la chaîne causale est lui-même une antiquité ; les accidents sont des défauts de *contrôle* dans une hiérarchie socio-technique entière, et si l'on retire l'événement déclencheur, un autre aurait servi[19]. L'objection est fondée — et à l'examen, elle radicalise ce chapitre au lieu de le réfuter. Car ce que Dekker et Leveson attaquent, c'est l'arrêt prématuré de la remontée : l'enquête qui s'arrête à « erreur humaine », au fusible, au composant cassé — c'est-à-dire très exactement le traitement symptomatique, monté d'un étage. Leur prescription — reconcevoir la structure de contrôle plutôt qu'exciser un événement — est la dissolution d'Ackoff appliquée aux systèmes socio-techniques. Le désaccord réel est étroit : il porte sur la terminaison de la remontée — un nœud, ou un réseau. La réponse est de définir le mot plutôt que de le défendre : la racine n'est pas une origine métaphysique, c'est **le niveau le plus profond où l'on peut encore agir**. Dans un accident industriel, ce niveau est souvent un réseau de conditions, et le singulier est une fiction commode. Dans un artefact conçu — un logiciel, une machine —, la régression se termine généralement sur une décision que quelqu'un a prise, et qu'on peut reprendre : l'architecture est le fond du puits. C'est le privilège du concepteur sur l'enquêteur d'accident : son système a un plan, et le plan a un auteur.

Le trajet du chapitre tient en une ligne : remonter du symptôme à la décision qui le produit, savoir reconnaître en route le bruit qui n'a pas de cause et la coïncidence qui n'en est pas une, et viser la dissolution plutôt que la victoire. Ce qui a été trouvé au fond du

puits, il faut maintenant l'écrire — en faire un problème posé, avec ses contraintes et ses critères. C'est l'objet du chapitre suivant.

---

1. Kaplan, A. (1964). *The Conduct of Inquiry : Methodology for Behavioral Science*. San Francisco : Chandler https://books.google.com/books/about/The_Conduct_of_Inquiry.html?id=_Kq_TXjfkj4C — le « principe de la recherche de l'ivrogne » (*the drunkard's search*). Le même ouvrage énonce la loi de l'instrument croisée au chapitre « Il n'y a pas de problème technique » : un seul livre, deux infirmités de l'enquête. ↩

2. Senge, P. M. (1990). *The Fifth Discipline : The Art and Practice of the Learning Organization*. New York : Doubleday/Currency https://www.worldcat.org/title/fifth-discipline-the-art-and-practice-of-the-learning-organization/oclc/21226996 — archétypes *Shifting the Burden* (la solution symptomatique atrophie la solution fondamentale) et *Fixes that Fail* (la correction court-termiste aggrave le problème qu'elle traite). ↩

3. Ohno, T. (1988). *Toyota Production System : Beyond Large-Scale Production*. Cambridge, MA : Productivity Press https://doi.org/10.4324/9780429273018 (éd. japonaise 1978), chap. 1, « Repeating Why Five Times » : « en répétant *pourquoi* cinq fois, la nature du problème comme sa solution deviennent claires. » ↩

4. Reason, J. (1990). *Human Error*. Cambridge : Cambridge University Press https://doi.org/10.1017/cbo9781139062367, notamment chap. 7 (« Latent errors and systems disasters ») — défaillances actives contre conditions latentes, les « pathogènes résidents » ; origine du modèle dit du gruyère. ↩

5. Fil « Why do CAT and filmic render clipped areas so weirdly », pixls.us, 2021 https://discuss.pixls.us/t/solved-why-do-cat-and-filmic-render-clipped-areas-so-weirdly/26426 — le titre attribue au module de rendu un écrêtage produit par le capteur et hérité de la chaîne entière ; le même fil est versé, pour d'autres raisons, au dossier du chapitre « Le conflit bien tempéré ». ↩

6. Hilton, D. J. & Slugoski, B. R. (1986). « Knowledge-based causal attribution : The abnormal conditions focus model ». *Psychological Review*, 93(1), 75-88 https://doi.org/10.1037/0033-295X.93.1.75 — l'attribution causale ordinaire isole la condition *anormale* sur un fond de conditions tenues pour normales, dans la lignée de Hart, H. L. A. & Honoré, T. (1985). *Causation in the Law*, 2e éd. Oxford : Clarendon Press. Même mécanisme par les contrefactuels : Kahneman, D. & Miller, D. T. (1986). « Norm Theory : Comparing Reality to Its Alternatives ». *Psychological Review*, 93(2), 136-153 https://doi.org/10.1037/0033-295X.93.2.136 — l'événement le plus facile à « défaire » par la pensée reçoit le blâme. ↩

7. Spellman, B. A. (1997). « Crediting Causality ». *Journal of Experimental Psychology : General*, 126(4), 323-348 https://doi.org/10.1037/0096-3445.126.4.323 — dans une chaîne de causes, chacune est créditée en proportion de ce qu'elle a *changé* à la probabilité du résultat, sachant ce qui la précédait : le dernier maillon qui fait la différence visible ramasse une part disproportionnée du blâme. ↩

8. John, L. K., Blunden, H. & Liu, H. (2019). « Shooting the Messenger ». *Journal of Experimental Psychology : General*, 148(4), 644-666 https://doi.org/10.1037/xge0000586 — 11 expériences : le porteur d'une mauvaise nouvelle est jugé moins aimable même quand il n'y est manifestement pour rien (tirage au sort), l'effet étant porté par le besoin de donner un agent à ce qui va mal. Voir déjà Rosen, S. & Tesser, A. (1970). « On Reluctance to Communicate Undesirable Information : The MUM Effect ». *Sociometry*, 33(3), 253-263 https://doi.org/10.2307/2786156. ↩

9. Cook, R. I. & Woods, D. D. (1994). « Operating at the Sharp End : The Complexity of Human Error », in Bogner, M. S. (dir.), *Human Error in Medicine*. Hillsdale, NJ : Erlbaum, 255-310 https://psycnet.apa.org/record/1994-98177-007 — le *sharp end* hérite des conditions latentes créées au *blunt end* (conception, organisation) et en absorbe le blâme, parce qu'il est le lieu visible de la défaillance ; la distinction active/latente est celle de Reason (1990), citée plus haut. ↩

10. Eisenstadt, M. (1997). « My Hairiest Bug War Stories ». *Communications of the ACM*, 40(4), 30-37

https://doi.org/10.1145/248448.248456 — 59 récits de débogage de professionnels : plus de la moitié de la difficulté tient au « gouffre temporel ou spatial » entre la cause racine et le symptôme. Le modèle général — défaut → infection → défaillance, la défaillance n'étant que l'extrémité observable d'une propagation — est chez Zeller, A. (2009). *Why Programs Fail : A Guide to Systematic Debugging*, 2e éd. Burlington, MA : Morgan Kaufmann, chap. 1. ↩

11. Deming, W. E. (1986). *Out of the Crisis*. Cambridge, MA : MIT Center for Advanced Engineering Study https://doi.org/10.7551/mitpress/11457.001.0001 — causes communes et causes spéciales (lignée de Shewhart, W. A., *Economic Control of Quality of Manufactured Product*, Van Nostrand, 1931 https://archive.org/details/in.ernet.dli.2015.150272) ; l'expérience de l'entonnoir, pp. 327-332. ↩

12. Einhorn, H. J. & Hogarth, R. M. (1986). « Judging Probable Cause ». *Psychological Bulletin*, 99(1), 3-19 https://doi.org/10.1037/0033-2909.99.1.3 — les « indices de causalité » (ordre temporel, contiguïté, similarité) sont employés même lorsqu'ils trompent. Lagnado, D. A. & Sloman, S. A. (2006). « Time as a Guide to Cause ». *Journal of Experimental Psychology : Learning, Memory, and Cognition*, 32(3), 451-460 https://doi.org/10.1037/0278-7393.32.3.451 — l'ordre temporel l'emporte sur la covariation. Buehner, M. J. & May, J. (2003). « Rethinking Temporal Contiguity and the Judgement of Causality ». *Quarterly Journal of Experimental Psychology A*, 56(5), 865-

890 https://doi.org/10.1080/02724980244000675 — la conviction causale décroît avec le délai. Synthèse : Matute, H. *et al.* (2015). « Illusions of Causality ». *Frontiers in Psychology*, 6, 888 https://doi.org/10.3389/fpsyg.2015.00888. ↩

13. Tversky, A. & Kahneman, D. (1971). « Belief in the Law of Small Numbers ». *Psychological Bulletin*, 76(2), 105-110 https://doi.org/10.1037/h0031322 ; Stanovich, K. E. & West, R. F. (2008). « On the Relative Independence of Thinking Biases and Cognitive Ability ». *Journal of Personality and Social Psychology*, 94(4), 672-695 https://doi.org/10.1037/0022-3514.94.4.672 ; Kahan, D. M., Peters, E., Dawson, E. C. & Slovic, P. (2017). « Motivated Numeracy and Enlightened Self-Government ». *Behavioural Public Policy*, 1(1), 54-86 https://doi.org/10.1017/bpp.2016.2 ; Kahneman, D. & Klein, G. (2009). « Conditions for Intuitive Expertise : A Failure to Disagree ». *American Psychologist*, 64(6), 515-526 https://doi.org/10.1037/a0016755. Sur les experts de terrain retournant leur propre verdict sous un contexte biaisant : Dror, I. E., Charlton, D. & Péron, A. E. (2006). « Contextual Information Renders Experts Vulnerable to Making Erroneous Identifications ». *Forensic Science International*, 156(1), 74-78 https://doi.org/10.1016/j.forsciint.2005.10.017. Je m'en tiens à « l'expérience n'immunise pas » : la littérature sur l'âge est mixte, et ne soutient pas que les décideurs plus âgés soient plus biaisés en général. ↩

14. Sur l'assurance : Price, P. C. & Stone, E. R. (2004). « Intuitive Evaluation of Likelihood Judgment Producers : Evidence for a Confidence Heuristic ». *Journal of Behavioral Decision Making*, 17(1), 39-57 https://doi.org/10.1002/bdm.460 ; nuance : Tenney, E. R., MacCoun, R. J., Spellman, B. A. & Hastie, R. (2007). « Calibration Trumps Confidence as a Basis for Witness Credibility ». *Psychological Science*, 18(1), 46-50 https://doi.org/10.1111/j.1467-9280.2007.01847.x — la confiance crédibilise par défaut, jusqu'à ce que ses erreurs de calibration deviennent visibles ; c'est précisément ce que la demi-journée perdue finit par rendre visible. Sur le titre, l'expérience fondatrice de la crédibilité de la source : Hovland, C. I. & Weiss, W. (1951). « The Influence of Source Credibility on Communication Effectiveness ». *Public Opinion Quarterly*, 15(4), 635-650 https://doi.org/10.1086/266350 — le même texte, attribué à une source de haute ou de basse crédibilité, change d'autant plus d'opinions qu'il est attribué à la première. Les deux axes sont indépendants : un profane peut être assuré, un expert prudent. ↩

15. LeBlanc, V. R., Brooks, L. R. & Norman, G. R. (2002). « Believing Is Seeing : The Influence of a Diagnostic Hypothesis on the Interpretation of Clinical Features ». *Academic Medicine*, 77(10 Suppl.), S67-S69 https://doi.org/10.1097/00001888-200210001-00022 ; Croskerry, P. (2003). « The Importance of Cognitive Errors in Diagnosis and Strategies to Minimize Them ». *Academic Medicine*, 78(8), 775-780 https://doi.org/10.1097/00001888-

200308000-00003 — origine du terme *diagnostic momentum* ; Graber, M. L., Franklin, N. & Gordon, R. (2005). « Diagnostic Error in Internal Medicine ». *Archives of Internal Medicine*, 165(13), 1493-1499 https://doi.org/10.1001/archinte.165.13.1493 — sur 100 erreurs, des facteurs cognitifs dans 74 %, l'arrêt prématuré de l'enquête (*premature closure*) en tête. Le pont avec la coïncidence : Mamede, S. *et al.* (2010). « Effect of Availability Bias and Reflective Reasoning on Diagnostic Accuracy Among Internal Medicine Residents ». *JAMA*, 304(11), 1198-1203 https://doi.org/10.1001/jama.2010.1276 — un cas récent similaire dégrade le diagnostic suivant ; la réflexion délibérée le répare. ↩

16. Nickerson, R. S. (1998). « Confirmation Bias : A Ubiquitous Phenomenon in Many Guises ». *Review of General Psychology*, 2(2), 175-220 https://doi.org/10.1037/1089-2680.2.2.175 ; Leventhal, L. M., Teasley, B. E. & Rohlman, D. S. (1994). « Analyses of Factors Related to Positive Test Bias in Software Testing ». *International Journal of Human-Computer Studies*, 41(5), 717-749 https://doi.org/10.1006/ijhc.1994.1079 ; Çalıklı, G. & Bener, A. (2013). « Influence of Confirmation Biases of Developers on Software Quality : An Empirical Study ». *Software Quality Journal*, 21(2), 377-416 https://doi.org/10.1007/s11219-012-9180-0. ↩

17. Bettenburg, N., Just, S., Schröter, A., Weiss, C., Premraj, R. & Zimmermann, T. (2008). « What Makes a Good Bug Report ? ». *Proceedings of the 16th ACM SIGSOFT International*

Symposium on Foundations of Software Engineering, 308-318 https://doi.org/10.1145/1453101.1453146 ; version étendue : Zimmermann, T. *et al.* (2010). *IEEE Transactions on Software Engineering*, 36(5), 618-643 https://doi.org/10.1109/TSE.2010.63 — 466 développeurs d'Apache, Eclipse et Mozilla classent les étapes de reproduction, les traces et les cas de test en tête de l'utile ; l'écart entre ce que les rapporteurs fournissent et ce que les développeurs attendent est l'objet même de l'étude. ↩

18. Ackoff, R. L. (1981). « The Art and Science of Mess Management ». *Interfaces*, 11(1), 20-26 https://doi.org/10.1287/inte.11.1.20 : *absolve, resolve, solve, dissolve* — « et la plus grande des quatre est la dissolution ». ↩

19. Dekker, S. (2014, 3e éd.). *The Field Guide to Understanding "Human Error"*. Farnham : Ashgate https://doi.org/10.1201/9781317031833 — la section « There Is No Root Cause ». Leveson, N. G. (2011). *Engineering a Safer World : Systems Thinking Applied to Safety*. Cambridge, MA : MIT Press https://doi.org/10.7551/mitpress/8179.001.0001 . Dans le même sens : Hollnagel, E. (2014). *Safety-I and Safety-II*. Farnham : Ashgate https://doi.org/10.1201/9781315607511 . Reason lui-même (1990) tenait déjà les conditions latentes pour plurielles et combinatoires — la « cause racine » au singulier est une vulgarisation de l'analyse causale, pas sa meilleure version. ↩

5. Poser le problème

Ce qui a été trouvé au fond du puits n'existe pas encore : tant que le problème n'est pas écrit, chacun en porte une version différente, et ces versions divergent en silence jusqu'au jour où elles se rencontrent — en général au milieu du développement, au pire moment. Le folklore des programmeurs le dit à sa manière : sans exigences ni conception, programmer est l'art d'ajouter des bugs à un fichier texte vide[1]. Écrire le problème n'est pas une formalité administrative ; c'est l'acte qui le fait exister comme objet commun — quelque chose qu'on peut se passer de main en main, contester, amender, et surtout clore. Le chapitre « Le mot et la chose » l'a établi : les problèmes de conception n'ont pas de règle d'arrêt, et la règle doit donc venir de la méthode. La voici. Le problème posé par écrit, avec ses critères de validation, *est* la règle d'arrêt.

Que faut-il écrire ? Tout ce dont la solution optimale dépend — et elle dépend de bien plus que de l'énoncé. À problème égal, la meilleure solution change selon la façon dont on le regarde, le budget dont on dispose, les priorités qu'on s'est données — y compris ses valeurs et sa politique (non, ce n'est pas un gros mot, et l'ingénieur politisé n'est pas plus dogmatique qu'un autre parce que ses principes osent s'afficher au grand jour) —, la personne qui utilisera la solution, et la durée de vie qu'on attend de cette solution. Cinq paramètres, et aucun n'est technique : ce sont des décisions, humaines, arbitraires, discutables, situées. Un

problème posé, c'est un énoncé plus ces 5 décisions, assumées par écrit, comme un contrat de ce qu'on s'engage à résoudre. Tant qu'elles restent implicites, chaque participant de la discussion les remplit avec les siennes, et l'on retombe dans les débats sans fin du chapitre « Le mot et la chose » — non parce que les gens sont de mauvaise foi, mais parce qu'ils ne résolvent pas le même problème sans le savoir et que l'effort de se mettre d'accord sur la définition de l'objet n'a pas été fait.

Le cœur de l'énoncé est le cas d'usage : une tâche définie, accomplie par un utilisateur défini, dans un temps défini. Retoucher un portrait posé, par une photographe de mariage, entre la remise des fichiers et la livraison de l'album. Les 3 termes sont nécessaires. Une tâche sans utilisateur est une abstraction ; un utilisateur sans tâche est une cible marketing ; sans horizon de temps, on ne peut ni dimensionner ni valider. Et s'il n'y a pas de cas d'usage, il n'y a pas de problème — donc rien à concevoir, et le mieux est de ne pas ouvrir l'éditeur de code. « Ça serait cool si… » n'est pas un cas d'usage ; c'est ainsi qu'on fabrique les collections de gadgets incohérents.

Il reste permis, et même nécessaire, de prototyper sans cas d'usage — pour explorer une idée, apprendre une technique, vérifier qu'une piste mérite ou non un cahier des charges. Mais cette activité porte un nom, et ce n'est pas concevoir : c'est de la recherche. La distinction n'est pas de vocabulaire, elle est de circuit. La recherche produit de la connaissance ; la conception produit un outil. La première a le droit d'échouer, de rester inachevée et de finir à la poubelle — c'est même son régime normal ; l'outil, lui, sera maintenu pendant des années entre les

mains de gens qui en dépendent. Le désastre commence quand le circuit se court-circuite : quand le prototype de recherche, faute d'avoir trouvé son cas d'usage, est livré quand même — versé directement dans les mains des utilisateurs finaux, avec ses hypothèses non vérifiées et son ergonomie d'éprouvette. La recherche ne doit jamais atteindre l'utilisateur final sans intermédiaire ; entre elle et lui, il y a précisément tout ce livre — le cas d'usage trouvé, le problème posé, les critères, la validation. « Ça serait cool si… » est un excellent début de recherche, et une catastrophe de feuille de route.

Vient ensuite la décision la plus évitée : le périmètre — pour qui, dans quelles conditions. Elle est évitée parce qu'elle exclut, et qu'exclure est inconfortable ; mais on n'a pas le choix : un outil qui prétendrait servir à tout le monde ne serait adapté à personne. Pour un outil de métier, on ne peut pas servir un professionnel et un débutant/amateur de la même manière. Ainsi, la formation et l'entraînement conditionnent jusqu'à la perception elle-même. Une étude sur l'évaluation visuelle en agroalimentaire — des jurys de dégustation jugeant la couleur de jus d'orange — a mesuré l'écart : après entraînement, les sujets distinguaient des différences de couleur presque deux fois plus fines qu'avant[2]. Ce que l'œil exercé voit, l'œil neuf ne le voit pas — littéralement. Concevoir le même outil pour le débutant et pour l'expert, ce n'est donc pas servir deux publics : c'est frustrer les deux, car l'un a besoin qu'on lui cache et qu'on lui simplifie ce que l'autre a besoin de contrôler. Ce qui est un détail secondaire pour l'un est une priorité pour l'autre. Et le métier ne suffit pas à définir le périmètre : le photoreporter qui transmet ses images

depuis l'événement, la photographe d'art qui travaille lentement pour un tirage d'exposition et l'équipe de mode qui vise la même qualité à cadence industrielle partagent le même métier de création d'images, mais n'ont pas les mêmes conditions d'usage. J'ai fini par l'écrire noir sur blanc pour mon propre logiciel : il attend des utilisateurs une connaissance intermédiaire à avancée de la théorie des couleurs et de la technique photographique, et ne vise pas les débutants. On peut contester ce choix ; on ne peut plus prétendre qu'il n'a pas été fait. Choisir pour qui l'on conçoit est une décision de valeurs — et il ne faut pas essayer de la faire passer pour un résultat de statistiques. Il faut même la regarder pour ce qu'elle est : choisir pour qui l'on conçoit, c'est choisir qui l'on renonce à servir, pour servir correctement les autres. La partie « La liberté et ses conditions » mesurera ce qu'une pièce socialement homogène tranche ainsi sans même s'apercevoir qu'elle tranche. Tout cela s'arbitre en nommant clairement, et honnêtement, qui l'on exclue de la solution.

Reste à écrire ce qui fera foi : les critères de validation. Un cahier des charges détaille les conditions que la solution devra remplir pour être acceptée, exprimées en grandeurs mesurables ou au moins objectives : le nombre d'opérations pour accomplir la tâche, le temps requis, la formation nécessaire avant de s'en servir, les standards avec lesquels il faudra s'interfacer, et les interdits — les postures et les gestes qui blessent, car le clavier et la souris infligent des lésions de travail répétitif aussi réelles que celles des machines-outils. Deux règles gouvernent ce document. La première vient du chapitre « Il n'y a pas de problème technique » : aucune technologie, aucun dispositif dans l'énoncé

— le cahier des charges dit ce qui doit être vrai à la fin, jamais par quel moyen il doit le devenir. La seconde est une règle d'ordre : les critères sont posés *avant* de chercher des solutions, parce que l'inverse ne se produit pas — une fois qu'on tient une solution qui nous plaît, on écrit les critères qu'elle remplit. Concevoir est un travail où il faut penser contre son propre cerveau, qui est paresseux et retombe de lui-même dans ses habitudes ; la séparation stricte entre le problème et la solution en est le garde-fou.

Le document change aussi la nature des conflits. Prenons un cas réel, vécu sur mon logiciel, dont l'interface a deux vues : la table lumineuse, où l'on trie les images, et la chambre noire, où on les retouche. Deux camps d'utilisateurs y exigeaient des choses incompatibles : les uns voulaient les outils de tri dupliqués dans la chambre noire, les autres des réglages d'image dupliqués dans la table lumineuse — chacun réclamant les fonctions de l'autre vue dans la sienne, et chaque duplication allongeant une interface déjà saturée. Aucune négociation de solutions ne peut aboutir dans ces termes : les demandes se contredisent par construction, l'écran n'est pas extensible, et la duplication fonctionnelle duplique aussi (au moins en partie) le volume de code, la dette technique et le coût de maintenance. Face à cette impasse, le réflexe moderne a un nom respectable, le compromis, et une réputation usurpée de vertu. Regardons ce qu'il est réellement : une résolution du conflit par le partage des pertes — chaque partie renonce à la moitié de ses exigences, et l'arrangement passe pour équitable parce que tout le monde y perd également. « J'accepte de souffrir puisque tu souffres aussi » : voilà la morale

du compromis, et il faut un drôle d'endoctrinement pour y voir une solution. Car l'arithmétique des besoins ne fait pas de demi-comptes : à moitié satisfait égale, le plus souvent, complètement frustré. Un soir, ma compagne veut aller au cinéma et je veux aller au restaurant. Le compromis serait d'écourter le dîner pour ne voir que la seconde moitié du film : une soirée gâchée pour deux, au prix fort — autant rester à la maison. Les manuels de logique connaissent la forme générale de l'erreur, l'*argumentum ad temperantiam* : ce n'est pas parce qu'une position est à mi-chemin entre deux autres qu'elle est juste — le milieu n'est qu'une position de plus, qui doit faire ses preuves comme les autres. En conception, le milieu de deux solutions n'est, par définition, optimal pour personne. On en revient au même problème que celui de l'utilisateur moyen.

Le compromis n'est pas une solution : c'est le symptôme d'une analyse des besoins interrompue trop tôt. Les théoriciens de la négociation ont montré où se joue le déblocage : non entre les *positions* — ce que chacun exige —, mais entre les *intérêts* — ce dont chacun a besoin[3]. Reprenons la soirée. Pourquoi le restaurant ? Parce que je suis trop fatigué pour cuisiner et faire la vaisselle. Pourquoi le cinéma ? Pour s'évader une heure ou deux dans un monde imaginaire. Personne n'a *besoin* du restaurant ni du cinéma en soi ; le besoin porte sur ce que ces lieux procurent — et une fois les besoins nus, ils ne se contredisent plus : nous voulons la même chose, nous détendre. Plats à emporter et film à la maison : les deux besoins sont servis en entier, pour moins cher que le compromis qui les aurait mutilés tous les deux. Il en va de même de nos deux vues, et la remontée y était moins évidente

que pour la soirée. Pourquoi vouloir trier dans la chambre noire ? Parce que basculer vers la table lumineuse coûtait plus d'une seconde de gel de l'interface, à chaque aller-retour, des centaines de fois par séance. Pourquoi régler dans la table lumineuse ? La même réponse, dans l'autre sens. Personne n'avait besoin des doublons : le besoin commun, que toutes les positions masquaient, était de *ne pas perdre de temps au passage d'une vue à l'autre*. La solution ne se trouvait donc dans aucune des demandes, ni dans leur partage : elle était sous l'interface, dans le moteur — rendre la bascule instantanée en évitant les recalculs, en passant par la mise en cache (encore lui). Cela fait, chaque vue a pu être spécialisée pour sa tâche et débarrassée de ses doublons : les deux camps servis en entier, par *moins* d'interface — une seule conception, sans variante ni mode, née de l'intérêt commun que les positions cachaient. C'est à la conception, pas à la négociation, de trouver cet objet-là. On reconnaît le mouvement du chapitre « La racine et le symptôme » : quand deux exigences semblent se contredire, c'est qu'on tient des ramifications, pas encore des racines — on n'a pas remonté assez haut. Le compromis est choisi par défaut parce que cette remontée coûte du temps et une compétence qui n'a rien de technique ; mais menée à terme, elle débouche sur des solutions qui gagnent sur les deux tableaux — et qui sont généralement plus simples, donc moins chères à maintenir.

Reste le cas résiduel : des besoins qui, après une remontée honnête, demeurent incompatibles. La réponse n'est toujours pas le milieu — c'est la séparation : deux produits, deux configurations, deux modes, chacun entier pour son public.

Quand les compromis deviennent inévitables en série, ce n'est pas un destin à accepter, c'est le signe qu'une solution unique essaie de couvrir deux problèmes distincts, et qu'il faut la scinder. Car empiler les compromis a un coût humain précis : condamner des gens à ne travailler qu'avec des solutions sous-optimales, à s'inventer des contournements qui s'impriment ensuite dans leurs habitudes — et quand ils se révoltent contre cette accumulation, on diagnostique, avec une pointe de condescendance, une « résistance au changement ». Il est parfaitement sain de se battre contre les compromis : accepter le compromis, c'est accepter la conception ratée comme une fatalité, accepter de perdre un peu de son confort et de sa productivité à chaque nouveau changement. Ça ne peut pas être une posture rationnelle, dans la durée.

Une fois le problème posé, la suite est un chemin balisé, que les écoles d'ingénieurs enseignent : séance d'idées d'abord — toutes notées, même les absurdes, car elles débloquent les associations —, puis tri sur la faisabilité, maquettes comparées, prototype, et confrontation du prototype aux tests de validation écrits dans le cahier des charges ; on itère ou l'on passe en production. Je ne détaille pas les techniques d'animation de ces séances — il existe des méthodes, aucune n'est sacrée. Ce qui est sacré, c'est l'ordre : le problème, les critères, puis seulement les idées. Une séance d'idées sans cahier des charges lu par tous n'est pas une méthode de conception, c'est un sondage d'humeurs.

Sous ce processus particulier court une logique générale de résolution de problèmes, plus vieille que le logiciel, et qui sert à toutes les échelles. George Polya l'a formalisée en 4

mouvements : comprendre le problème, concevoir un plan, l'exécuter, puis revenir sur ses pas pour examiner la solution obtenue[4]. Trois outils s'y ajoutent, que l'ingénierie pratique sans toujours les nommer. La décomposition, du global au particulier : on découpe le problème en sous-problèmes dont chacun est plus simple que le tout, et l'on remonte régulièrement d'un étage pour vérifier qu'on résout encore le bon problème — les techniques vivent en couches, chaque couche absorbe qui s'y attarde, et le but se perd si l'on ne fait pas ce va-et-vient. La localisation avant la résolution : on ne résout pas un problème qu'on n'a pas situé, et la méthode reine est la dichotomie — couper l'espace de recherche en deux, tester de quel côté vit le défaut, recommencer ; c'est ainsi qu'on isole une panne, une régression dans un historique de commits, une hypothèse fautive dans un raisonnement. Et la variation contrôlée : ne changer qu'un facteur à la fois, sans quoi l'effet observé n'est attribuable à rien — le diagnostic différentiel du médecin, la boucle d'hypothèses du chapitre « Compter les absents ». Cette logique est sans contenu : elle fonctionne sur un bug, un cahier des charges, une organisation. C'est pourquoi les chapitres de ce livre se ressemblent d'une échelle à l'autre — ce sont les mêmes mouvements, appliqués à des objets différents.

Pour tenir les objectifs eux-mêmes, une grille m'a servi de point de départ : la méthode SMART, formulée par George Doran en 1981 pour l'écriture d'objectifs de gestion[5]. Un objectif bien posé est *spécifique*, *mesurable*, *assignable*, *réaliste* et *borné dans le temps*. Spécifique : une tâche concrète, qu'on prend et qu'on termine — pas une intention globale, et certainement rien qui se formule dans la langue vague des politiciens. Le contre-exemple

canonique, dans mon domaine : « transformons Darktable en Lightroom ». Perdu d'avance, non par manque d'ambition, mais parce que rien n'y est spécifique — ni le logiciel, ni son histoire, ni son architecture, ni les habitudes de ses utilisateurs ne sont ceux du concurrent ; il faut partir d'où l'on est et nommer l'amélioration précise qu'on vise. Mesurable : ce sont les critères de validation — le point le plus mal compris, auquel le chapitre suivant est consacré. Assignable : quelqu'un porte l'objectif, nommément ; un objectif que personne ne porte est un vœu. Réaliste : proportionné aux ressources qu'on a réellement, y compris pour la maintenance — beaucoup de bonnes idées meurent là, non d'être mauvaises, mais de n'avoir jamais eu les moyens de leur durée. Borné dans le temps : un chantier sans échéance ni budget de temps est une dérive annoncée. La grille n'a rien de sacré non plus — Doran lui-même écrivait que tous les objectifs ne méritent pas les 5 critères. Elle ne garantit aucun résultat ; elle force seulement à réfléchir à ce qu'on fait avant de le faire, et c'est déjà rare.

L'objection contemporaine à tout ceci a un nom : le développement agile. Elle tient en 3 griefs. Le cahier des charges appartiendrait au « cycle en cascade », cette méthode ancienne où l'on spécifie tout le projet d'abord, où l'on développe ensuite, et où l'on livre à la fin — deux ans plus tard, un produit conforme au document et inadapté au besoin, qui a changé entre-temps. On apprendrait davantage en livrant tôt et souvent qu'en écrivant des documents. Et la conception devrait donc *émerger* des itérations, au lieu d'être arrêtée d'avance[6]. L'objection mérite mieux qu'un rejet, car son constat est exact : la spécification

intégrale et figée échoue, et l'itération courte est effectivement le meilleur instrument de mesure qu'on ait — livrer, c'est mesurer. Mais il faut regarder ce que les praticiens agiles font, plutôt que ce que le slogan dit : une *user story* est un cas d'usage — un utilisateur défini, une tâche définie, un bénéfice défini —, et sa *definition of done* est une liste de critères de validation posés avant le développement. L'agilité n'a pas aboli le cahier des charges ; elle l'a découpé en tranches et rapproché de l'exécution — ce qui est une amélioration du format, pas une dispense du contenu. Ce que l'itération fait émerger, ce sont les solutions ; le problème, lui, n'émerge pas : quelqu'un le pose, tranche par tranche, ou bien personne ne le pose — et itérer sans problème posé porte un autre nom, la dérive. Vitesse et direction ne sont pas la même grandeur.

L'histoire a tranché plus vite que prévu — et ce ne sont pas les nostalgiques du cycle en cascade qui ont mené la charge, ce sont les auteurs du manifeste agile eux-mêmes. Dave Thomas, l'un des 17 signataires, dès 2014 : le mot « agile » a été « subverti au point d'être à peu près vide de sens », la communauté qui s'en réclame est devenue « une arène où consultants et vendeurs écoulent services et produits », le mot lui-même « un terme de marketing, coopté pour faire vendre, comme *éco* et *naturel* »[7]. Andy Hunt, signataire : l'agile a dégénéré en récitation de règles, porté par des « zélotes — au sens où le zélote est celui qui redouble d'efforts après avoir oublié son but », et « les méthodes agiles elles-mêmes n'ont pas été agiles : voilà une ironie »[7]. Martin Fowler, signataire : l'état ordinaire du terrain est le *faux-agile* — « le nom, mais rien des pratiques ni des valeurs » —, et « le complexe agilo-

industriel qui impose des méthodes aux gens est une absolue parodie » : l'inversion terme à terme du premier énoncé du manifeste, les personnes au-dessus des processus[7]. Ron Jeffries, signataire : « cela me brise le cœur de voir les idées que nous avons écrites dans le manifeste servir à rendre la vie des développeurs pire, et non meilleure » — et il appelle les développeurs à abandonner l'agile nommé en gardant ses valeurs[7]. Quant à la mise « à l'échelle » vendue aux grandes organisations, le co-créateur de Scrum l'a accueillie d'une phrase : « les gars de RUP sont de retour », avec consultants de personnalisation, « exactement comme RUP »[7]. Le verdict académique range le dossier : Bertrand Meyer a trié le paradigme en bon et en laid — au bon, l'accueil du changement, les itérations courtes, le code qui marche, les tests centralisés ; au laid, très précisément l'abolition de l'amont : le refus d'exigences écrites avant de coder, les *user stories* comme substitut d'exigences, les tests comme substitut de spécification, le rejet du travail d'architecture a priori[8]. Relisez ces deux listes : ce qui survit de l'agilité est exactement ce que ce chapitre en a gardé, et ce qui la discrédite est exactement ce qu'il refuse.

Que retenir de cette trajectoire ? Qu'elle est exemplaire, et le chapitre « Le culte du cargo » en donnera la loi générale : une révolte contre le processus — 17 praticiens dressés contre la lourdeur documentaire — s'est achevée en la plus lourde des liturgies : cérémonies quotidiennes, certifications à étages, rôles dédiés, cadres « à l'échelle » — des danses de la pluie contre l'incertitude irréductible du développement, exécutées avec d'autant plus de ferveur qu'elles maîtrisent moins l'incertitude.

Ce ne sont pas les idées qui ont échoué : itérer court, livrer pour mesurer, découper la spécification — tout cela tient, et ce chapitre le garde. C'est le rite qui a dévoré la raison, comme il la dévore chaque fois qu'une pratique se transmet plus vite que ses pourquoi. Une méthode de conception n'échappe pas à la discipline qu'elle impose aux autres : elle aussi doit dire son cas d'usage, ses critères — et accepter d'être jugée dessus.

Le problème est posé, le périmètre choisi, les critères écrits. Tout repose maintenant sur un mot laissé en suspens : *mesurable*. C'est le maillon où la discipline peut devenir imposture — car on peut mesurer beaucoup, et se mentir avec des chiffres mieux qu'avec des mots. Mesurer sans se mentir est l'objet du chapitre suivant.

---

1. Attribué à Louis Srygley — « Without requirements or design, programming is the art of adding bugs to an empty text file » —, sans source primaire vérifiable : c'est un aphorisme de folklore professionnel, cité ici comme tel. ↩

2. Fernández-Vázquez, R., Stinco, C. M., Hernanz, D. *et al.* (2013). « Colour training and colour differences thresholds in orange juice ». *Food Quality and Preference*, 30(2), 320-327. https://doi.org/10.1016/j.foodqual.2013.05.018 Seuil de différence perceptible (ΔE CIE 1976) : environ 2,8 avant entraînement, 1,5 après. ↩

3. Fisher, R. & Ury, W. (1981). *Getting to Yes : Negotiating Agreement Without Giving In*. Boston : Houghton Mifflin https://openlibrary.org/books/OL4260851M/Getting_to_yes.

Trad. fr. *Comment réussir une négociation*. Paris : Seuil. Le principe central : négocier sur les intérêts, jamais sur les positions. ↩

4. Polya, G. (1945). *How to Solve It*. Princeton : Princeton University Press https://doi.org/10.1515/9781400828678 (trad. fr. *Comment poser et résoudre un problème*, Paris : Dunod, 1965) — comprendre, planifier, exécuter, examiner : la grille canonique, écrite pour les mathématiques et transposable à tout problème posé. ↩

5. Doran, G. T. (1981). « There's a S.M.A.R.T. Way to Write Management's Goals and Objectives ». *Management Review*, 70(11), 35-36. Lettres originales : *Specific, Measurable, Assignable, Realistic, Time-related*. Les variantes courantes (*atteignable*, *achievable*, *ambitieux*…) sont des mutations postérieures de l'acronyme. ↩

6. Beck, K. *et al.* (2001). « Manifesto for Agile Software Development » (https://agilemanifesto.org/) ; Beck, K. (1999). *Extreme Programming Explained : Embrace Change*. Reading, MA : Addison-Wesley https://books.google.com/books/about/Extreme_Programming_Explained.html?id=G8EL4H4vf7UC. Le manifeste oppose « du logiciel qui fonctionne » à « une documentation exhaustive » — tout en maintenant, dans ses principes, la conversation continue avec le métier et l'attention constante à la conception. ↩

7. Thomas, D. (2014). « Agile is Dead (Long Live Agility) ». pragdave.me, 4 mars 2014

https://pragdave.me/thoughts/active/2014-03-04-time-to-kill-agile.html — « The word “agile” has been subverted to the point where it is effectively meaningless » ; « It became a marketing term, coopted to improve sales in the same way that words such as eco and natural are ». Hunt, A. (2015). « The Failure of Agile ». toolshed.com, 6 mai 2015 https://toolshed.com/2015/05/the-failure-of-agile.html — « agile methods themselves have not been agile. Now there’s an irony for you ». Fowler, M. (2018). « The State of Agile Software in 2018 ». martinfowler.com, 25 août 2018 https://martinfowler.com/articles/agile-aus-2018.html (transcription éditée de son intervention à Agile Australia, Melbourne, juin 2018) — « The Agile Industrial Complex imposing methods on people is an absolute travesty ». Jeffries, R. (2018). « Developers Should Abandon Agile ». ronjeffries.com, 10 mai 2018 https://ronjeffries.com/articles/018-01ff/abandon-1/ . Schwaber, K. (2013). « unSAFe at any speed ». kenschwaber.wordpress.com, 6 août 2013 https://kenschwaber.wordpress.com/2013/08/06/unsafe-at-any-speed/ — « The boys from RUP (Rational Unified Process) are back ». Les cinq sont cosignataires du manifeste de 2001 (Schwaber est aussi co-créateur de Scrum) ; toutes les citations sont nos traductions. ↩ ↩ ↩ ↩ ↩

8. Meyer, B. (2014). *Agile ! The Good, the Hype and the Ugly.* Cham : Springer https://doi.org/10.1007/978-3-319-05155-0 — le chapitre 3 s’intitule « The Enemy : Big Upfront Anything », et le chapitre 11 dresse le tri ; le « laid » y comprend le refus des exigences écrites en amont, les *user*

Stories comme substitut d'exigences abstraites, les tests comme substitut de spécification et le rejet du travail d'architecture a priori. Même auteur que l'« ouvert/fermé » du chapitre « Le culte du cargo ». ↩

6. Mesurer sans se mentir

Le chapitre précédent a laissé les critères de validation entre les mains d'un mot : *mesurable*. C'est le maillon le plus solide de la méthode et le plus facile à corrompre, car les chiffres jouissent d'un privilège : on ne les soupçonne pas. Une opinion s'affiche comme opinion ; un nombre se présente comme un fait, même quand il a été fabriqué, choisi ou détourné. Il y a deux grandes manières de se mentir avec des mesures : mesurer une chose en croyant en caractériser une autre, et transformer la mesure en objectif. Et en amont des deux, un fait premier que l'usage des tableaux de bord fait oublier : un chiffre n'existe pas à l'état naturel — quelqu'un l'a calculé, par une méthode et sur des données qu'il a choisies, et souvent parmi un ensemble de méthodes possibles ayant chacune leurs forces, leurs faiblesses, leur contexte, et plus ou moins adaptées aux conditions du problème présent.

Un chiffre est un produit fini, et il faut le lire comme tel. Avant d'arriver sous vos yeux, il a traversé deux étages de décisions. Le premier est la méthode : une équation, une procédure d'agrégation, une façon de formuler le problème, des seuils, des hypothèses simplificatrices qui permettent le calcul — autant de choix qui ne sont pas neutres et qui décident du résultat autant que la réalité mesurée. Un taux de plantage se calcule par session ou par utilisateur, et le même logiciel en sort stable ou branlant ; un temps moyen d'accomplissement de tâche inclut ou exclut les

tentatives abandonnées, et la même interface en sort fluide ou décourageante ; un score de satisfaction additionne des questions que quelqu'un a choisies, formulées et pondérées. La métrologie — la science de la mesure — ne dit pas autre chose : un résultat de mesure n'a de sens qu'accompagné de sa méthode et de son incertitude, et la méthode se publie précisément parce qu'elle se conteste. C'est le régime normal de la mesure physique ; le tableau de bord logiciel, lui, enterre trop souvent sa méthode dans une requête que personne ne relit. Comprendre la méthode n'est donc pas un zèle d'expert, c'est la condition pour savoir ce qu'on lit — et la discuter est un droit : contester une équation, ou le modèle qu'elle représente (ou infère), n'est pas contester un fait. Le second étage est la donnée : elle a été collectée quelque part, par quelqu'un, sous un protocole qui a ses propres filtres — qui a été interrogé, qu'est-ce qui a été journalisé, qu'est-ce qui a été silencieusement perdu, comment ont été posées les questions. Ces biais ne se voient pas toujours dans les données, parce que certains n'y laissent aucune trace : la session qui plante n'envoie pas son rapport, l'utilisateur excédé désinstalle avant le questionnaire. Les deux étages se combinent en une conclusion qu'il faut garder devant chaque tableau de bord : quand on lit un chiffre, on lit déjà l'opinion de quelqu'un — sur la bonne façon de calculer, et sur les données qui méritaient d'être recueillies. On comprend du même coup d'où le chiffre tire son aura : il paraît puissant et objectif exactement dans la mesure où l'on ignore comment il a été obtenu. Et cette ignorance est bien gardée, car la peur des mathématiques — si communément partagée, si tôt installée — garantit que presque personne n'osera demander à voir le calcul. L'autorité du nombre ne vient pas de la rigueur de

sa fabrication ; elle vient de ce que sa fabrication est invisible et que son langage intimide. Ceux qui font métier de mesurer sont, de tous, les moins impressionnés par un nombre nu. Cela ne disqualifie pas les chiffres ; cela leur applique le régime du chapitre « Écouter sans obéir » : un chiffre est un témoignage, qui se laisse interroger — qui l'a calculé, comment, sur quoi — et un chiffre qu'on ne peut pas auditer n'est pas une mesure, c'est un argument d'autorité en habit d'objectivité.

Ceci posé, à quoi sert une métrique ? Elle sert à comparer deux solutions sur un même critère, et par là à clore une discussion : entre deux propositions, celle qui accomplit la tâche en moins d'étapes, en moins de temps, avec moins d'erreurs, gagne — et l'on passe à la suite. C'est tout, et c'est immense : la comparaison outillée est précisément la règle d'arrêt que le chapitre « Le mot et la chose » réclamait. Ce qu'une métrique ne doit jamais devenir, c'est un objectif d'optimisation. La formule est connue sous le nom de loi de Goodhart : dès qu'une mesure devient une cible, elle cesse d'être une bonne mesure[1]. Le mécanisme est sans mystère — dès qu'un chiffre commande la récompense, les gens optimisent le chiffre, pas ce qu'il était censé refléter — et l'exemple canonique vient de Donald Campbell, qui étudiait les indicateurs de l'action publique : faites du taux de résolution d'enquêtes l'indicateur de performance des policiers, et vous obtiendrez — entre autres — des policiers qui dissuadent les gens de porter plainte quand l'affaire s'annonce compliquée[2]. Le taux monte, la réalité qu'il mesurait se dégrade, et la dégradation est invisible puisque l'indicateur est devenu aveugle par construction. Pire que de ne pas mesurer : mesurer une chose en

croyant en caractériser une autre, car on prend alors de mauvaises décisions avec toute l'assurance de l'objectivité.

Voyons le mécanisme au travail sur la métrique favorite des discussions d'interface — celle que les ergonomes ont formalisée dès 1980 sous le nom de *keystroke-level model*, qui prédit le temps d'une tâche en additionnant ses gestes élémentaires[3] : le nombre d'opérations nécessaires pour arriver au résultat. Elle a tout pour plaire — objectivable, comparable, pas une opinion. Elle capte pourtant une dimension sur 3, et les comptes complets inversent souvent le verdict. Première dimension manquante : la sécurité. Darktable s'enorgueillit d'un flux de travail « sans clic », où l'essentiel des opérations se déclenche au survol de la souris sur le *widget* graphique — moins d'opérations, en effet. On peut aussi fabriquer une voiture sans clé de contact ni ceinture de sécurité : cela retire des étapes au démarrage ; si elle tue davantage de passagers, est-ce un meilleur design ? Dans l'ancienne table lumineuse, déplacer la souris pouvait modifier les métadonnées d'une image qu'on n'avait pas choisie — la note qu'on croyait donner à une photo allait à celle que le curseur effleurait, et l'image disparaissait du filtrage courant sans qu'on comprenne pourquoi. J'en ai tiré deux règles, qui coûtent des clics et qui les valent : aucune écriture de données sans sélection explicite, et une sélection n'est explicite que par un geste dur — un clic, une touche ; le survol est réservé à la lecture. La friction devant une opération destructrice n'est pas un défaut d'ergonomie, c'est un équipement de sécurité — ce que la conception industrielle appelle une *fonction de forçage*, et l'industrie japonaise un *poka-yoke*, un dispositif anti-erreur qui rend le geste dangereux

impossible ou coûteux[4]. Ces deux règles ne protègent pas l'utilisateur attentif, qui n'en a pas besoin : elles protègent le pressé, le fatigué, le débutant — celui qui ne pourra pas se défendre contre ma conception. Deuxième dimension manquante : ce que j'appelle le *prix d'entrée cognitif* — découvrir l'outil, comprendre ce qu'on manipule, savoir prédire ce qu'un geste va faire. Il ne se compte pas en clics. L'utilisateur installé depuis 10 ans se moque que l'agencement soit sensé ou absurde : il en connaît les recoins et veut aller vite ; la vraie question est de savoir si l'organisation a du sens pour celui qui arrive, parce que c'est lui qui doit comprendre sans connaître. Troisième pièce du dossier : les comptes complets de la centralisation des masques, promis au chapitre « Écouter sans obéir ». Le reproche — « c'est plus loin qu'avant » — comptait les clics et le déplacement de curseur, et il comptait juste. Mais il comptait les clics de l'ancien design comme s'ils étaient gratuits, et ils ne l'étaient pas. Les commandes de masque, logées au bas de modules déjà hauts, apparaissaient sous le bord de l'écran — les statistiques d'usage donnaient un premier décile d'écrans à 960 × 540 pixels utiles, et pour une bonne moitié des utilisateurs, il fallait de toute façon cliquer ou défiler pour les atteindre. Le défilement était d'ailleurs lui-même piégé : dans Darktable, la molette au-dessus d'un module fait défiler les *valeurs des curseurs*, pas le contenu du panneau — le geste « gratuit » pouvait modifier des réglages au passage. Et même ce prix payé, l'écran ne montrait toujours pas ensemble le module masqué et ses paramètres de masquage. Le design actuel inverse la répartition des coûts : la visibilité des commandes de masque est garantie entière, contre des clics et du trajet de souris en plus, et l'éloignement entre le module et son

masque. S'ajoute la ligne de cohérence : le mécanisme de masquage étant identique pour tous les modules, une fonction commune se centralise au lieu de se dupliquer — c'est, du reste, le choix de Lightroom. Quatre lignes de compte — les clics, le défilement piégé, la visibilité, la cohérence — là où la métrique en voyait une. Et le verdict garde la forme que ce chapitre impose aux verdicts de mesure : comparatif, jamais définitif. Je ne suis pas attaché à ce design ; de toutes les solutions auxquelles j'ai réfléchi, c'est la moins mauvaise, et le dossier reste ouvert à qui pense pouvoir produire un meilleur design.

Un dernier compte avant les instruments, car il montre ce que vaut le ressenti quand aucune mesure ne l'encadre. Il m'est arrivé plusieurs fois d'optimiser une pièce atomique du logiciel — une boucle, un cache, un transfert mémoire — et de le dire dans les notes de version. Dans les jours suivants, des utilisateurs venaient rapporter que d'autres parties du programme, sans rapport avec la pièce optimisée et qu'aucune ligne du changement ne touchait, étaient plus rapides elles aussi. Aucun de ces rapports ne s'appuyait sur un chronomètre ; tous étaient sincères ; et tous étaient contents. J'appelle cela le *sentiment de vitesse* : donnez aux gens le préjugé que le logiciel devrait être plus rapide, et ils le sentiront plus rapide, quels que soient les temps mesurés. Ce n'est pas une naïveté propre à mes utilisateurs, c'est une propriété de la perception. Le temps ressenti se manipule sans toucher au temps réel : une barre de progression de même durée, animée autrement, est jugée jusqu'à 11 % plus rapide[5]. L'annonce suffit : des participants à qui l'on dit qu'un écran fonctionne à une fréquence plus élevée qu'en

réalité jugent leur performance meilleure, à temps de réaction rigoureusement égal — et la même perception fantôme s'obtient en annonçant une intelligence artificielle ou une adaptation qui n'existent pas[6]. Le mécanisme est général, et il est d'ordre temporel : une information reçue *avant* l'expérience modifie l'expérience elle-même, pas seulement son compte rendu ; reçue après, elle ne change rien[7]. C'est exactement le régime des notes de version : on les lit avant de lancer le logiciel, et l'on y apprend ce qu'on va sentir. Sur l'ensemble de la littérature, d'ailleurs, l'efficacité mesurée, l'efficience mesurée et la satisfaction déclarée ne se corrèlent que faiblement[8]. La conclusion n'est pas que le ressenti ne compte pas ; c'est qu'il ne *mesure* pas : il mesure l'attente, à laquelle la performance a contribué pour une part inconnue. Un gain de vitesse se prouve au banc, sur une tâche fixée, avant et après, par la personne qui a fait le changement — et l'éloge du premier jour, comme la protestation du premier jour dont le chapitre « Décider, puis tenir » fera l'anatomie, est un réflexe, pas un témoignage.

Reste la tentation symétrique de la foi dans les chiffres : décréter que l'essentiel — la satisfaction, le plaisir d'usage — ne se mesure pas, et s'épargner ainsi la mesure. La tentation est répandue chez les concepteurs d'outils experts, et j'en connais l'attrait : elle a pour elle l'évidence apparente. Mais la psychométrie de l'utilisabilité a produit des instruments, et il faut les regarder en face avant de les récuser. La norme ISO 9241-11 définit depuis 1998 l'utilisabilité comme le degré selon lequel un système peut être utilisé, *par des utilisateurs spécifiés*, pour atteindre *des buts spécifiés*, avec efficacité, efficience et satisfaction, *dans un contexte*

d'usage spécifié[9]. L'échelle SUS — 10 affirmations notées de 1 à 5, un score sur 100 — existe depuis 1986, a servi dans des milliers d'études, et possède ce qui fait un instrument sérieux : des normes (la moyenne empirique se situe vers 68, qui vaut donc 50e percentile — un score SUS n'est pas un pourcentage, et le « OK » spontané des utilisateurs correspond à un score *inférieur* à la moyenne) et une fiabilité mesurée[10]. Le NASA-TLX mesure depuis 1988 la charge mentale subjective sur 6 dimensions — c'est très exactement l'instrument standardisé de mon « prix d'entrée cognitif », validé de l'aviation à la chirurgie[11]. Il existe même une échelle pour le plaisir : l'AttrakDiff distingue la qualité *pragmatique* d'un produit de sa qualité *hédonique* — ce que l'usage procure, stimulation et identité —, et montre que les deux varient indépendamment[12]. Le champ teste ses propres outils et exécute ses mauvaises métriques.

Ce qui survit de l'objection est plus étroit et plus solide. Premièrement — et c'est le point central —, **aucune de ces échelles ne mesure la qualité de ce que l'outil a produit**. SUS mesure l'utilisabilité *perçue*; TLX, la charge *ressentie*; AttrakDiff, l'attrait *éprouvé*. Aucune ne dit si la photographie est bonne. L'« efficacité » d'ISO 9241 — l'exactitude et la complétude du résultat — suppose un critère de correction externe, qui n'existe sous forme standardisée pour aucune production esthétique : c'est le trou au centre de l'édifice, et il n'est pas près d'être comblé. Deuxièmement, tout cela est de l'autodéclaration de perception, et le chapitre « Écouter sans obéir » a donné le chiffre : la préférence déclarée ne prédit qu'un quart de la performance mesurée — la perception n'est pas la performance.

Troisièmement, les scores ne se lisent que relatifs à une norme et à un contexte ; la norme ISO le dit elle-même : l'utilisabilité n'est pas un attribut du produit, mais le résultat d'une interaction dans un contexte spécifié. Un SUS recueilli auprès de débutants ne dit à peu près rien d'un outil de métier dans des mains formées — on retrouve le périmètre du chapitre « Poser le problème », et l'on comprend pourquoi les comparaisons de logiciels experts par panels de novices produisent les classements qu'elles produisent. Quatrièmement, un score unique compresse des grandeurs incommensurables — SUS mélange la facilité d'apprentissage et l'efficience en usage, que tout oppose dans un outil de métier. La version honnête de l'objection n'est donc pas « ces échelles n'existent pas », mais : **ces échelles mesurent autre chose que ce qu'il faut juger pour un outil de travail** — elles mesurent l'expérience de l'utilisateur, jamais l'œuvre de l'utilisateur. Qu'on s'en serve pour ce qu'elles mesurent : comparer deux versions du même outil, sur la population visée par le cahier des charges, avec le score en instrument de comparaison — jamais en cible, Goodhart y veille — et le jugement sur l'œuvre reste à construire par d'autres moyens : les critères de validation du chapitre « Poser le problème », et la physique quand elle a son mot à dire.

Mesurer sans se mentir tient donc en peu de règles. Demander la méthode et la provenance des données avant de croire le chiffre — un nombre inauditable ne prouve rien. Savoir ce que la métrique mesure — et le redemander régulièrement, car la réponse change quand l'usage change. Comparer, ne jamais optimiser. Compléter chaque compte avant de rendre le verdict : la sécurité et le prix d'entrée ne se voient dans aucun décompte de clics. Concéder

aux instruments ce qu'ils mesurent, et leur refuser ce qu'ils ne mesurent pas. Reste une question que ce chapitre a laissée ouverte : tout chiffre a été collecté sur *quelqu'un* — un échantillon, jamais la population —, et la différence entre un écart significatif et du bruit, et la différence entre ceux qu'on compte et ceux qui manquent sont des questions de statistique. Elles méritent leur chapitre : compter les absents.

---

1. Goodhart, C. A. E. (1975). « Problems of Monetary Management : The U.K. Experience », in *Papers in Monetary Economics*, vol. I. Sydney : Reserve Bank of Australia https://doi.org/10.1007/978-1-349-17295-5_4 . La formulation célèbre — « quand une mesure devient une cible, elle cesse d'être une bonne mesure » — est de Strathern, M. (1997). « "Improving ratings" : audit in the British University system ». *European Review*, 5(3), 305-321 https://doi.org/10.1002/(SICI)1234-981X(199707)5:3%3C305::AID-EURO184%3E3.0.CO;2-4 . ↩

2. Campbell, D. T. (1979). « Assessing the impact of planned social change ». *Evaluation and Program Planning*, 2(1), 67-90 https://doi.org/10.1016/0149-7189(79)90048-x : plus un indicateur social quantitatif est utilisé pour la décision, plus il est soumis à des pressions de corruption et plus il corrompt le processus qu'il devait surveiller — avec, explicitement, l'exemple des statistiques policières. ↩

3. Card, S. K., Moran, T. P. & Newell, A. (1980). « The Keystroke-Level Model for User Performance Time with

Interactive Systems ». *Communications of the ACM*, 23(7), 396-410 https://doi.org/10.1145/358886.358895 ; développé dans *The Psychology of Human-Computer Interaction*, Hillsdale : Erlbaum, 1983 https://doi.org/10.1201/9780203736166 — la formalisation exacte du décompte de gestes que ce chapitre critique. ↩

4. Les *forcing functions* : Norman, D. A. (2013). *The Design of Everyday Things*, éd. révisée, chap. 4 (verrouillages, interverrouillages, blocages) https://www.hachettebookgroup.com/titles/don-norman/the-design-of-everyday-things/9780465050659/ . Le *poka-yoke* : Shingo, S. (1986). *Zero Quality Control : Source Inspection and the Poka-Yoke System*. Cambridge, MA : Productivity Press https://doi.org/10.4324/9780203733639 — l'anti-erreur à la source, dans l'industrie. ↩

5. Harrison, C., Yeo, Z. & Hudson, S. E. (2010). « Faster Progress Bars : Manipulating Perceived Duration with Visual Augmentations ». *Proceedings of CHI 2010*, 1545-1548 https://doi.org/10.1145/1753326.1753556 ; Harrison, C., Amento, B., Kuznetsov, S. & Bell, R. (2007). « Rethinking the Progress Bar ». *Proceedings of UIST 2007*, 115-118 https://doi.org/10.1145/1294211.1294231 — à durée identique, la fonction de progression et l'animation changent la durée perçue. Dans le même sens, sur la valeur plutôt que la vitesse : Buell, R. W. & Norton, M. I. (2011). « The Labor Illusion ». *Management Science*, 57(9), 1564-1579 https://doi.org/10.1287/mnsc.1110.1376 . ↩

6. Bosch, E., Welsch, R., Ayach, T., Katins, C. & Kosch, T. (2024). « The Illusion of Performance : The Effect of Phantom Display Refresh Rates on User Expectations and Reaction Times ». *CHI EA '24*, 1-6 https://doi.org/10.1145/3613905.3650875 ; Kosch, T., Welsch, R., Chuang, L. & Schmidt, A. (2022). « The Placebo Effect of Artificial Intelligence in Human-Computer Interaction ». *ACM Transactions on Computer-Human Interaction*, 29(6), art. 56 https://doi.org/10.1145/3529225 ; Denisova, A. & Cairns, P. (2015). « The Placebo Effect in Digital Games : Phantom Perception of Adaptive Artificial Intelligence ». *Proceedings of CHI PLAY 2015*, 23-33 https://doi.org/10.1145/2793107.2793109. ↩

7. Lee, L., Frederick, S. & Ariely, D. (2006). « Try It, You'll Like It ». *Psychological Science*, 17(12), 1054-1058 https://doi.org/10.1111/j.1467-9280.2006.01829.x — l'information donnée avant la dégustation change l'expérience ; donnée après, non. Le corrélat neural : Plassmann, H., O'Doherty, J., Shiv, B. & Rangel, A. (2008). « Marketing Actions Can Modulate Neural Representations of Experienced Pleasantness ». *PNAS*, 105(3), 1050-1054 https://doi.org/10.1073/pnas.0706929105 ; et sur la performance réelle, pas seulement perçue : Shiv, B., Carmon, Z. & Ariely, D. (2005). « Placebo Effects of Marketing Actions ». *Journal of Marketing Research*, 42(4), 383-393 https://doi.org/10.1509/jmkr.2005.42.4.383. ↩

8. Hornbæk, K. & Law, E. L.-C. (2007). « Meta-analysis of Correlations Among Usability Measures ». *Proceedings of*

CHI 2007, 617-626 https://doi.org/10.1145/1240624.1240722 — sur 73 études, efficacité, efficience et satisfaction ne se corrèlent qu'à r ≈ 0,2-0,3. ↩

9. ISO 9241-11:2018, *Ergonomie de l'interaction homme-système — Partie 11 : utilisabilité : définitions et concepts* https://www.iso.org/standard/63500.html, § 3.1.1 ; l'efficacité (3.1.12), l'efficience (3.1.13) et la satisfaction (3.1.14) y sont définies séparément. Première version : 1998. ↩

10. Brooke, J. (1996). « SUS : A "quick and dirty" usability scale », in Jordan, P. W. *et al.* (dir.), *Usability Evaluation in Industry*. Londres : Taylor & Francis https://doi.org/10.1201/9781498710411-35. Normes et fiabilité (α ≈ 0,91 sur 2 324 questionnaires) : Bangor, A., Kortum, P. T. & Miller, J. T. (2008). « An Empirical Evaluation of the System Usability Scale ». *International Journal of Human-Computer Interaction*, 24(6), 574-594 https://doi.org/10.1080/10447310802205776 ; percentiles et équivalents adjectivaux : Sauro, J. & Lewis, J. R. (2012). *Quantifying the User Experience*. Waltham, MA : Morgan Kaufmann https://www.sciencedirect.com/book/9780123849687/quantifying-the-user-experience. ↩

11. Hart, S. G. & Staveland, L. E. (1988). « Development of NASA-TLX : Results of Empirical and Theoretical Research », in Hancock, P. A. & Meshkati, N. (dir.), *Human Mental Workload*. Amsterdam : North-Holland, 139-183 https://doi.org/10.1016/s0166-4115(08)62386-9. Bilan à vingt ans : Hart, S. G. (2006). « NASA-Task Load Index (NASA-

TLX) ; 20 Years Later ». *Proceedings of the HFES Annual Meeting*, 50(9), 904-908 https://doi.org/10.1177/154193120605000909 . ↩

12. Hassenzahl, M., Burmester, M. & Koller, F. (2003). « AttrakDiff : Ein Fragebogen zur Messung wahrgenommener hedonischer und pragmatischer Qualität », in *Mensch & Computer 2003*. Stuttgart : Teubner, 187-196 https://doi.org/10.1007/978-3-322-80058-9_19 . ↩

7. Compter les absents

Le chapitre précédent s'est arrêté au bord d'une question : tout chiffre a été collecté sur quelqu'un. Ce quelqu'un est un échantillon — jamais la population entière —, et l'échantillon pose deux questions distinctes, que ce chapitre prend dans l'ordre : comment résumer ceux qui sont dans les chiffres ; et comment penser à ceux qui n'y sont pas. La seconde est la plus difficile, parce que les absents ne laissent aucune trace dans les données — c'est leur définition — et qu'ils sont pourtant, souvent, la ligne la plus importante du tableau.

Commençons par les présents. Le premier réflexe de résumé est la moyenne, et c'est presque toujours le mauvais : la moyenne ne dit quelque chose que d'une distribution à peu près gaussienne, et une poignée de valeurs extrêmes suffit à la fausser. Sur la quantité de mémoire vive de mes utilisateurs, une vingtaine de machines à 64 Go décalent la moyenne de tout l'échantillon — le chiffre obtenu ne décrit plus personne. Ce qu'on cherche à travers la moyenne, c'est en réalité autre chose : où se situe l'essentiel des utilisateurs. La réponse tient en 3 nombres — la médiane, le premier décile (P10) et le dernier (P90) : entre P10 et P90, on tient 80 % des utilisateurs, et c'est sur eux que se prennent les décisions. Car c'est cela, une décision technique lue statistiquement : un choix de qui elle inclut et de qui elle exclut. « L'écran de référence sera 1920 × 1080 » ne dit pas une préférence : il dit combien de machines passent sous la barre, et le

P10 donne le chiffre. Dans un autre registre, la distribution rend aussi le service promis au chapitre « La racine et le symptôme » : elle montre à quoi ressemble la variation normale. Avant de chercher la cause d'un écart, on regarde s'il sort de l'étendue ordinaire — tout ce qui se passe entre la moyenne ± n écarts-types est le bruit propre du système, celui qui en sort mérite un *pourquoi*[1].

Ensuite, l'échantillon n'est pas la population, et il n'en est jamais un extrait neutre : il a été filtré, et les filtres sont rarement ceux qu'on croit. Le logiciel libre filtre d'abord sur la compétence informatique, et il faut des chiffres pour mesurer combien ce filtre est brutal. L'enquête PIAAC de l'OCDE — 215 942 adultes testés dans 33 pays — donne l'échelle : environ un quart des adultes ne peut pas se servir d'un ordinateur du tout, les deux tiers ne dépassent pas les tâches à une étape, et 5 % seulement atteignent le niveau des tâches complexes[2] ; l'Union internationale des télécommunications compte, sur 91 pays, à peu près la moitié de la population incapable de télécharger et d'installer un logiciel[3]. Demander à quelqu'un d'ouvrir une ligne de commande, c'est donc avoir déjà écarté plus de 90 % du monde — avant la première question du sondage. Mes propres enquêtes portent la marque de ce filtre : parmi les utilisateurs de Darktable, plus de 35 % détiennent un master ou un doctorat, contre moins de 14 % en population générale[4]. La conséquence méthodologique est du chapitre « Mesurer sans se mentir » : un sondage de satisfaction aveugle, qui ne demande pas *qui* répond, ne mesure pas la satisfaction — il mesure qui utilise le logiciel. Les résultats disent autant la sociologie de l'échantillon que les qualités de l'outil ; il

faut des études à deux dimensions, la satisfaction *et* le qui, sans quoi l'on prend le profil de ses survivants pour un verdict sur son travail.

Le mot est lâché : survivants. Le piège le plus vicieux de la statistique d'usage est que, par construction, vos données ne contiennent que ceux qui utilisent le logiciel — jamais ceux qui ont renoncé, jamais ceux qui ne sont pas venus. Pire, parmi les utilisateurs, ceux qui vont coopérer aux sondages sont encore la fraction la plus motivée et émotionnellement impliquée dans le projet logiciel. Le raisonnement à leur sujet doit s'inverser, et l'inversion a un précédent célèbre. En 1943, le statisticien Abraham Wald, travaillant pour l'aviation américaine, développa une méthode pour estimer la vulnérabilité des avions à partir des dommages des seuls appareils *revenus* de mission : puisque les abattus manquent aux données, les zones où les survivants ne sont jamais touchés sont précisément celles où les coups abattent — c'est là qu'il faut blinder, pas là où les survivants portent leurs impacts[5]. L'anecdote telle qu'on la raconte — Wald retournant d'une phrase les officiers qui voulaient blinder les zones criblées — est une reconstruction tardive ; la méthode, elle, est documentée dans ses mémorandums, et c'est elle qui compte. Sa traduction logicielle est directe : à force d'améliorer l'outil pour ceux qui sont restés, on l'optimise pour le profil qui a survécu au filtre — et on le rend plus rebutant encore pour tous les autres. Le retour des utilisateurs vit dans la même bulle : ceux qui détestaient sont partis, ceux qui restent aiment l'outil comme il est, et leur unanimité ne prouve pas que l'outil est bon — elle

prouve que le filtre a fini son travail et qu'on a atteint une sorte d'homéostasie.

Les absents ont une autre fabrique, plus active : la prophétie autoréalisatrice, décrite par Robert Merton — une définition fausse de la situation qui suscite les comportements qui vont la rendre vraie après un certain temps[6]. En logiciel, elle a un cycle bien rodé : une fonctionnalité mal conçue est peu utilisée ; ses chiffres d'usage, faibles, justifient de ne jamais l'améliorer ; elle reste mal conçue, donc peu utilisée — et les chiffres ont raison pour de mauvaises raisons. Mes statistiques en donnent un cas d'école : 37 % de mes utilisateurs possèdent une tablette graphique, 6 % s'en servent dans le logiciel. Mais encore, 10 % possèdent un contrôleur MIDI, et 2 % s'en servent dans le logiciel. Sachant que les contrôleurs MIDI sont plutôt dédiés à la musique, et que le support dans Darktable est aussi complet qu'on peut l'imaginer le 2 % d'usage traduirait plutôt une absence d'intérêt. Sachant que la tablette graphique est un appareil spécialement dédié au traitement d'image et que Darktable utilise des masques dessinés, le 6 % d'usage traduirait plutôt une utilisabilité défaillante. Dans les deux cas, la fréquence d'utilisation ramenée à la fréquence de possession (1 pour 5 et 1 pour 6) donne déjà un indicateur plus complet que la simple utilisation, mais reste insuffisant pour conclure, et il faut donc investiguer spécialement le cas "possède sans utiliser". D'où la règle générale, qui est la posture scientifique élémentaire : un chiffre ne se contemple pas, il se confronte. On part d'une hypothèse — donc d'un préjugé, assumé comme tel —, on en déduit une prédiction — « si c'est vrai, on devrait observer ceci » —, et on regarde si les données

confirment ou infirment[7]. Collecter des statistiques suppose d'avoir déjà une idée de ce qu'on cherche à savoir ; un tableau de bord qu'on fixe sans question n'apprend rien, il rassure.

L'instrument lui-même a mauvaise réputation, et il faut d'abord le relativiser. La télémétrie basée sur le consentement volontaire (*opt-in*) penche vers les configurations des utilisateurs les plus engagés — les seuls qui laissent la collecte active. C'est l'exact biais du survivant : ils décrivent fidèlement qui est resté, et rien de qui est parti. J'en tire la hiérarchie qui vaut pour tout tableau de bord d'usage : la télémétrie est fiable pour *hiérarchiser à l'intérieur* de la population servie — quel format, quelle résolution, quelle carte graphique y pèse le plus —, jamais pour en fixer *la frontière* ; décider qui l'outil doit servir est l'affaire d'une comparaison avec une population de référence, pas d'un décompte de ceux qui sont déjà là. Ceci posé, j'ai doté mon logiciel de statistiques d'usage, et la première règle fut la transparence : dire ce qui est collecté, pourquoi, et publier ce qu'on en apprend — les utilisateurs acceptent peut-être plus facilement de participer quand ils voient à quoi servent les comptes. Le rendement est réel. Savoir que plus de 43 % des machines n'ont aucune accélération graphique suffit à refroidir l'enthousiasme pour les architectures qui misent tout sur la carte graphique : pour ces 43 %, un tel logiciel ne serait pas une option. La télémétrie hiérarchise la pile des problèmes au lieu de la subir, et elle protège aussi le mainteneur — quand une exigence bruyante représente 5 % de l'échantillon, le chiffre remet la discussion à sa taille. Cet usage défensif a sa limite, et elle est exactement la prophétie de Merton : « vous êtes 5 % » peut être la

mesure d'un besoin marginal ou celle d'un défaut qui a chassé les autres — le taux de possession, la comparaison avec une population de référence, l'écart entre avoir et se servir, sont les contrôles qui départagent. L'instrument ne dispense jamais de l'hypothèse.

Reste une objection de principe, qui a pour elle l'air du temps, et il faut l'énoncer en entier pour qu'elle ait sa chance. La voici : le jugement du concepteur est faillible, biaisé, invérifiable, alors collectons les données d'usage et faisons ce qu'elles disent : on garde ce qui est utilisé, on retire ce qui ne l'est pas, on priorise ce qui est demandé, et les débats se tranchent au tableau de bord plutôt qu'à l'autorité. C'est la doctrine dite *data-driven*, si chère aux technocrates : les données décident, l'humain exécute. Le chapitre entier lui a déjà répondu pièce par pièce : les données ne contiennent que les survivants d'un filtre qu'elles ne documentent pas ; elles se lisent à travers des hypothèses qu'elles ne fournissent pas ; et la décision de qui compte est un choix de valeurs qu'aucune donnée ne prendra — c'est le périmètre du chapitre « Poser le problème ». Parfois, la décision intelligente va contre les chiffres. Une catégorie qui pèse 2 % dans l'échantillon peut peser 15 ou 20 % dans la réalité, si elle a été filtrée, notamment par la complexité — l'installation difficile, le dépôt à configurer, l'antivirus qui déclare l'exécutable infecté. Ces absents-là manquent pour une mauvaise raison, et leur tendre la main est une décision qui n'aura jamais de majorité dans les données, puisqu'ils n'y sont pas. Si les gens sont absents parce que l'outil ne les intéresse pas, très bien ; s'ils sont absents parce qu'ils n'ont rien compris, c'est un problème de conception — et il

ne figurera dans aucun tableau de bord. Remarquer un besoin dont personne ne témoigne est le geste le plus coûteux du métier : rien ne le récompense, aucun chiffre ne le réclame, et c'est pourtant par lui que tout commence. Être guidé par les données vaut mieux que d'être guidé par son nombril, mais ce n'est pas une dispense de jugement. Les données informent la décision ; elles ne la prennent pas. Et toute prise de décision impose une hiérarchie de valeurs qui impose une hiérarchie de priorités : toutes doivent être clairement dites et assumées.

Résumer les présents par leur distribution, connaître le filtre de son échantillon, raisonner les absents par inversion, casser les prophéties avant qu'elles ne s'accomplissent, confronter chaque chiffre à une hypothèse : voilà l'outillage. Il reste alors le geste que tout cet appareil prépare et qu'aucun appareil n'accomplit — trancher, puis tenir ce qu'on a tranché. C'est l'objet du chapitre suivant.

---

1. Règle de poche, à manier comme telle : l'analyse rigoureuse de la stabilité d'un processus utilise les cartes de contrôle de Shewhart et Deming — limites à ±3 écarts- types autour de la moyenne d'un processus *suivi dans le temps*, pas les déciles d'une distribution instantanée (Shewhart, W. A. (1931). *Economic Control of Quality of Manufactured Product*. New York : Van Nostrand https://archive.org/details/in.ernet.dli.2015.150272 ; Deming, W. E. (1986). *Out of the Crisis*. Cambridge, MA : MIT CAES https://doi.org/10.7551/mitpress/11457.001.0001). L'étendue P10-P90 en donne une approximation utile pour des données

d'usage sans historique, mais elle décrit la dispersion, pas la stabilité : un système peut dériver tout entier en gardant la même étendue. ↩

2. OCDE (2016). *Skills Matter : Further Results from the Survey of Adult Skills*. Paris : Éditions OCDE https://doi.org/10.1787/9789264258051-en — volet « résolution de problèmes dans des environnements technologiques » (PIAAC ; 215 942 adultes, 33 pays) : 26 % des adultes ne peuvent pas utiliser un ordinateur, 69 % ne dépassent pas le niveau 1 (tâches à une étape), 5 % atteignent le niveau 3. Synthèse : Nielsen, J. (2016). « The Distribution of Users' Computer Skills ». Nielsen Norman Group https://www.nngroup.com/articles/computer-skill-levels/ . ↩

3. Union internationale des télécommunications (2021). *Measuring digital development : Facts and Figures 2021*. Genève : UIT https://www.itu.int/itu-d/reports/statistics/facts-figures-2021/ — sur 91 pays, 48,8 % de la population ne sait pas télécharger et installer un logiciel. ↩

4. Enquêtes menées par l'auteur auprès des utilisateurs de Darktable en 2020 (n = 1 101) https://eng.aurelienpierre.com/2023/01/who-are-the-Darktable-users-in-2020/ et 2022 (n = 1 253) https://eng.aurelienpierre.com/2026/07/who-are-the-Darktable-users-in-2022/ ; données anonymisées et code de traitement publiés avec les résultats (eng.aurelienpierre.com). Population générale : OCDE, niveaux de diplôme des 25-64 ans. ↩

5. Wald, A. (1943). *A Method of Estimating Plane Vulnerability Based on Damage of Survivors*. Statistical Research Group, mémorandums 85-126 ; rééd. CNA, 1980 https://www.cna.org/analyses/1980/a-method-of-estimating-plane-vulnerability . Analyse moderne : Mangel, M. & Samaniego, F. J. (1984). « Abraham Wald's Work on Aircraft Survivability ». *JASA*, 79(386), 259-267 https://doi.org/10.1080/01621459.1984.10478038 . Sur la part de légende du récit populaire : Casselman, B. (2016). « The Legend of Abraham Wald ». AMS Feature Column https://www.ams.org/publicoutreach/feature-column/fc-2016-06 — la scène et le célèbre diagramme aux impacts rouges n'apparaissent nulle part dans les mémorandums. ↩

6. Merton, R. K. (1948). « The Self-Fulfilling Prophecy ». *Antioch Review*, 8(2), 193-210 https://doi.org/10.2307/4609267 . ↩

7. Popper, K. (1934). *Logik der Forschung*. Vienne : Springer ; trad. angl. *The Logic of Scientific Discovery*. Londres : Hutchinson, 1959 https://doi.org/10.4324/9780203994627 — la connaissance avance par conjectures exposées à la réfutation, jamais par contemplation des données. ↩

8. Décider, puis tenir

Tout l'appareil des chapitres précédents — le problème posé, les critères écrits, les mesures faites — prépare un geste qu'aucun appareil n'accomplit : trancher. La délibération a une forme, que quiconque a conçu en groupe reconnaîtra : on choisit une solution, on la discute, elle paraît moins bonne ; on en choisit une autre, on la discute, elle pâlit à son tour ; et de proche en proche on revient à la première idée. Ce circuit n'est pas un échec, c'est la topologie normale d'un espace de solutions où rien n'est parfait — mais il faut savoir le reconnaître : quand on a fait 3 fois le tour de la question, la discussion n'apprend plus rien, et il faut trancher. C'est ici que l'objectivation du chapitre « Mesurer sans se mentir » rend son service — les critères mesurables permettent de clore ce que les goûts laisseraient ouvert —, à une condition d'hygiène : ne pas détourner une métrique de ce qu'elle veut dire pour se donner raison. Et la barre de décision est plus basse qu'on ne croit : la conception étant itérative, une décision n'a pas à être optimale immédiatement — elle a à être meilleure que l'état présent, et réversible si elle ne l'est pas.

Trancher n'est pourtant que la moitié du geste. L'autre moitié commence le lendemain, car en matière d'outils, la décision se teste en vrai — rien ne remplace l'usage — et le test grandeur nature a un coût social que tout mainteneur connaît : chaque changement provoque des protestations, quelle que soit sa qualité. Ce phénomène a un nom, l'aversion au changement, et

une littérature qui en démonte les mécanismes. Le biais de statu quo : à options égales, on préfère l'existant, simplement parce qu'il existe[1]. L'aversion à la perte : un changement d'interface *retire* quelque chose qu'on avait — et les pertes pèsent psychologiquement plus lourd que les gains équivalents[2]. L'effet de simple exposition : le familier bénéficie d'un avantage d'agrément sur le nouveau, jusqu'à ce que le nouveau ait accumulé sa propre familiarité[3]. Et un coût bien réel : changer d'outil invalide des automatismes acquis, et le coût de réapprentissage est à peu près proportionnel au savoir-faire perdu[4]. Quatre mécanismes, tous indépendants de la qualité du nouveau design. La conséquence est brutale pour qui veut juger un changement : la protestation du premier jour ne dit à peu près rien. Les cas d'école le confirment. Quand Facebook lança son fil d'actualité en 2006, la protestation toucha une fraction notable des utilisateurs — pendant que l'usage, lui, montait : la préférence révélée contredisait l'aversion déclarée. Quand Microsoft remplaça les menus d'Office par le ruban, l'accueil fut globalement favorable — *sauf* chez les utilisateurs les plus expérimentés de l'ancienne interface[5] : la résistance se concentre là où le capital d'habitude est le plus grand, pas là où le design est le plus mauvais.

J'ai vécu ce théorème en petit. Le jour où j'ai retiré la boîte de recherche des modules — remplacée par une recherche contextuelle au raccourci clavier : Ctrl+P, puis le nom du module —, les protestations furent générales. Or il fallait de toute façon taper le nom du module : on passait déjà par le clavier, donc connecter la recherche à un raccourci clavier n'imposait aucun

passage supplémentaire de la souris au clavier. Les utilisateurs n'avaient rien perdu — même fonction, présentée autrement — et l'interface libérait une ligne en hauteur ; la fonction restait pleinement découvrable, puisque le raccourci est annoncé et la recherche accessible : ce qu'on avait retiré n'était pas la capacité de chercher, mais une *boîte* permanente qui encombrait l'interface. Le temps de l'expliquer deux ou trois fois, et tout le monde a fini par juger la solution plus efficace ; mais il a fallu tenir, quelques jours, contre des protestations venues — comme pour le ruban — des plus rodés, c'est-à-dire de ceux qui avaient le plus d'automatismes à réapprendre et le moins à gagner. Si j'avais cédé le premier soir, la bonne solution n'aurait jamais vu le jour.

Comment, alors, distinguer la protestation d'habitude — qui passe — du vrai défaut de conception — qui reste ? Il existe un critère, et il est d'une élégance remarquable : **l'aversion au changement ne peut pas exister chez qui n'a pas connu l'ancien**[6]. Un nouvel utilisateur n'a ni statu quo à défendre, ni perte à ressentir, ni automatisme à réapprendre : s'il bute, ce n'est pas l'habitude — c'est le design. Le diagnostic s'ensuit : la difficulté que les utilisateurs *sans histoire* reproduisent, ou celle qui persiste chez les anciens après une vraie réaccoutumance, est un défaut réel ; la protestation qui décroît jusqu'à un niveau de satisfaction meilleur qu'avant n'était que la friction du réapprentissage. Et la réaccoutumance a des ordres de grandeur : quelques jours d'usage honnête avant qu'une opinion soit recevable — le temps que la pénalité de simple exposition se dissipe —, quelques semaines avant que le verdict soit définitif ; la ré-automatisation complète des gestes est plus lente encore — la formation d'un

automatisme plafonne en médiane à une soixantaine de jours[7]. D'où le protocole qui rend le test socialement possible, et qui est un contrat : les utilisateurs accordent au changement un essai réel et assez long — pas un survol indigné du premier soir —; le concepteur promet le retour arrière si l'expérience n'est pas concluante, et tient cette promesse. Le retour arrière n'est pas une humiliation, c'est une clause du contrat — et c'est elle qui rend l'essai exigible. Ce n'est pas non plus une courtoisie de plus : promettre le retour arrière, c'est reconnaître qu'on expérimente sur le travail de quelqu'un d'autre. Rendre ses choix réversibles tant qu'on n'a pas la preuve qu'ils font du bien à l'utilisateur n'est pas une prudence de tempérament, c'est un devoir.

Deux pièges attendent les sorties de ce protocole. Le premier est le sophisme des coûts irrécupérables : conserver une mauvaise solution *parce qu'*on y a passé 3 mois. Le temps dépensé est perdu quoi qu'on décide; la seule question ouverte est le coût d'après, et une mauvaise solution maintenue en fait perdre davantage encore[8]. S'entêter n'est pas de la constance, c'est payer deux fois. Le second piège est symétrique : ne rien oser, sous prétexte que cela devrait marcher du premier coup. On ne crée rien sans la liberté de se tromper. Un logiciel de production n'est pas un laboratoire à ciel ouvert, mais il exige une latitude d'expérimentation — bornée par le protocole ci-dessus, précisément pour qu'elle reste exigible. Et l'inverse du refus du changement n'est pas la fuite en avant : changer pour le plaisir du neuf, du disruptif, de l'innovant, c'est infliger les 4 mécanismes de l'aversion à ses utilisateurs sans contrepartie. Le changement

se justifie par un compte — chapitre « Mesurer sans se mentir » —, jamais par sa date.

Reste le dernier mot du titre : tenir. Une décision close doit le rester parce qu'une décision qui se rouvre à volonté n'en est pas une, et que les discussions perpétuellement rouvertes sont le trou sans fond que le chapitre « Le mot et la chose » cherchait à éviter. La règle de réouverture est la même que la règle de clôture : du nouveau — une donnée nouvelle, un cas d'usage nouveau, un défaut que le protocole a confirmé — rouvre le dossier ; une opinion de plus, non. Face à la même discussion qui revient pour la énième fois sans élément nouveau, quand l'explication a été donnée, le protocole proposé, l'essai refusé, il est légitime de fermer — au titre de l'expérience et de la connaissance du dossier, et au titre du travail que l'on fournit et que l'interlocuteur ne fournit pas. Ce n'est pas la manière élégante ; c'est le disjoncteur, et un disjoncteur ne sert qu'en dernier recours — utilisé en routine, il ferait de l'autorité un mode de gouvernement, ce que la dernière partie de ce livre combat précisément. Entre l'autoritarisme et la timidité, la ligne de crête est étroite ; elle se tient en publiant ses critères, en promettant ses retours arrière, et en réservant l'usage de la force pour les impasses.

L'objection finale est connue d'avance : « l'utilisateur a toujours raison ». Le livre a déjà réparti les droits — l'utilisateur a autorité sur ses douleurs, jamais sur les solutions (chapitre « Écouter sans obéir ») —, et ce chapitre ajoute la pièce temporelle : la protestation du premier jour n'est pas un témoignage d'usage, c'est un réflexe de perte ; le témoignage commence après l'essai honnête, et le fil d'actualité de Facebook comme le ruban d'Office

montrent ce que vaut l'aversion déclarée face à la préférence révélée. L'utilisateur a toujours raison sur ce qui lui fait mal ; il a rarement raison, le premier soir, sur ce qui vient de changer.

Poser, mesurer, compter, décider, tenir : 5 gestes, et aucun n'a demandé qu'on invente un outil sur mesure — chacun s'est appuyé sur un instrument déjà rodé ailleurs, emprunté à qui l'avait rodé avant nous. Il en reste quelques-uns dans la boîte, qu'aucun chapitre n'avait de raison propre de convoquer, et qu'il faut pourtant connaître avant de refermer cette partie du livre — car une discipline qui ignore son outillage improvise deux fois plus qu'elle ne devrait. C'est l'objet du chapitre suivant.

---

1. Samuelson, W. & Zeckhauser, R. (1988). « Status Quo Bias in Decision Making ». *Journal of Risk and Uncertainty*, 1(1), 7-59. https://doi.org/10.1007/bf00055564 ↩

2. Kahneman, D. & Tversky, A. (1979). « Prospect Theory : An Analysis of Decision under Risk ». *Econometrica*, 47(2), 263-291 https://doi.org/10.2307/1914185 — l'asymétrie pertes/gains (aversion à la perte), appliquée au changement d'interface : le nouveau « retire ce qu'on avait ». ↩

3. Zajonc, R. B. (1968). « Attitudinal Effects of Mere Exposure ». *Journal of Personality and Social Psychology*, 9(2, 2e partie), 1-27. https://doi.org/10.1037/h0025848 ↩

4. Singley, M. K. & Anderson, J. R. (1989). *The Transfer of Cognitive Skill*. Cambridge, MA : Harvard University Press https://openlibrary.org/works/OL4639862W — le coût de

transfert croît avec la part des automatismes invalidés : réel, temporaire, proportionnel au recouvrement perdu. ↩

5. Fil d'actualité Facebook (septembre 2006) : épisode documenté par la presse de l'époque — protestation massive et très visible, usage en hausse. Ruban Office : Dostál, M. (2010). « User Acceptance of the Microsoft Ribbon User Interface ». *Proceedings DNCOCO 2010*, 143-149. https://wseas.us/e-library/conferences/2010/Faro/DNCOCO/DNCOCO-25.pdf ↩

6. Sedley, A. & Müller, H. (2013). « Minimizing Change Aversion for the Google Drive Launch ». *CHI '13 Extended Abstracts*, 2351-2354 https://doi.org/10.1145/2468356.2468767 — définit l'aversion au changement, en suit la trajectoire jusqu'à une « nouvelle base de référence », et pose le diagnostic différentiel : « si le produit est très difficile à apprendre ou à utiliser [pour un nouvel utilisateur], ce ne peut pas être de l'aversion au changement — ce sont des défauts fondamentaux de conception. » ↩

7. Lally, P., van Jaarsveld, C. H. M., Potts, H. W. W. & Wardle, J. (2010). « How are habits formed : Modelling habit formation in the real world ». *European Journal of Social Psychology*, 40(6), 998-1009 https://doi.org/10.1002/ejsp.674 — plateau d'automaticité atteint en 66 jours en médiane (étendue : 18 à 254). Les durées d'essai données ici sont une pratique de terrain, pas un seuil issu de la littérature. ↩

8. Arkes, H. R. & Blumer, C. (1985). « The Psychology of Sunk Cost ». *Organizational Behavior and Human Decision Processes*, 35(1), 124-140. DOI ↩

9. S'outiller sans se soumettre

Les chapitres qui précèdent ont posé une discipline : poser le problème, mesurer sans se mentir, compter les absents, décider puis tenir. Aucun de ces gestes n'a été inventé ici, et c'est une bonne nouvelle : la conception est une méthode enseignée depuis un demi-siècle, avec ses manuels, ses outils et ses résultats mesurés. La mauvaise nouvelle est que presque personne, dans le logiciel libre, ne les a ouverts. On y conçoit comme on bricole le samedi après-midi : à l'intuition, à l'enthousiasme, au dernier qui a parlé. Ce chapitre fait donc l'inventaire de la boîte à outils — non pas tout ce qui existe, mais ce qu'un concepteur d'outils de travail devrait connaître avant de prétendre faire mieux. Et il le fait sous une règle que le chapitre « Mesurer sans se mentir » impose : chaque outil sera pesé à son niveau de preuve. Certains tiennent sur des essais cliniques ; d'autres sur une expérience de laboratoire ; d'autres encore ne sont que du vocabulaire commode drapé d'autorité. Les confondre, c'est déjà se soumettre.

Commençons par le socle, car il faut savoir qu'il existe : la conception systématique est un corpus académique constitué. L'école allemande l'a codifiée dès les années 1970 — Pahl et Beitz, la « bible » de la conception mécanique outre-Rhin, découpe tout projet en 4 phases : clarifier la tâche, concevoir le principe, incarner la solution, définir le détail[1]. L'école américaine en a fait le manuel standard des écoles d'ingénierie et de commerce — Ulrich et Eppinger, 8 éditions depuis 1995 : identifier les besoins,

établir les spécifications, générer plusieurs concepts, les sélectionner contre des critères, puis seulement développer[2]. Qu'on relise cette partie du livre avec ces tables des matières en main : poser le problème, c'est la clarification de la tâche ; mesurer et compter, ce sont les spécifications et leurs critères ; décider puis tenir, c'est la sélection de concepts. Ce livre n'invente pas sa discipline — il la retrouve, et le fait mérite d'être dit crûment : quand un projet de logiciel improvise sa conception sur un fil de discussion, dans un forum, il n'est pas en avance sur ces méthodes « d'industriel », il a un demi-siècle de retard sur elles.

De ce corpus, le premier outil à retenir est la matrice de sélection de concepts, dite matrice de Pugh[3]. Le principe : on choisit un concept de référence — le *datum*, souvent la solution existante —, on aligne les concepts candidats en colonnes et les critères du cahier des charges en lignes, et chaque candidat est jugé, critère par critère, mieux (+), pareil (S) ou moins bien (–) que la référence. Elle est presque toujours déformée : la méthode originale ne pondère rien et ne calcule rien — les versions à coefficients sont des greffes tardives. Sa force n'est pas de produire un score, c'est d'organiser une convergence : au tour suivant, on hybride les candidats, on prend pour référence celui qui s'impose, et on recommence. La matrice ne décide pas à la place de l'équipe ; elle force la discussion à passer par chaque critère, un par un, face à une référence commune — exactement ce que le brainstorming d'opinions ne fait jamais. C'est un outil de pensée structurée, pas une machine à objectivité, et quiconque vous vend le contraire vend la déformation, pas la méthode.

Le deuxième outil a, lui, des morts évitées à son actif : la checklist. Un réanimateur a réduit une procédure connue de tous — la pose de cathéters centraux — à une liste de 5 gestes élémentaires, tous archi-connus, tous régulièrement sautés ; testée dans son hôpital puis déployée dans les réanimations de tout un État américain, la liste a fait s'effondrer les occurrences d'infections qu'on croyait être le prix inévitable du soin[4]. L'Organisation mondiale de la santé a généralisé : 19 points de contrôle autour d'une opération chirurgicale, essayés dans 8 villes sur 4 continents — mortalité passée de 1,5 % à 0,8 %, complications majeures de 11 % à 7 %[4]. Le mécanisme mérite d'être compris, car il n'a rien de bureaucratique : une checklist est de la mémoire institutionnelle sortie des têtes — la liste des défaillances connues, tenue à jour, consultée au moment où l'attention humaine est le moins fiable. Et le garde-fou est déjà écrit au chapitre « Mesurer sans se mentir » : la liste alerte, elle n'absout pas. Une checklist dont on coche les cases pour être en règle, au lieu de chercher ce qu'elle a détecté, est morte — le chapitre « L'outil transparent » en donnera un exemple chiffré.

Le troisième outil s'attaque au silence poli qui entoure les décisions, et le chapitre « Décider, puis tenir » a montré ce que ce silence coûte. Le *prémortem* renverse l'exercice de revue : au lieu de demander « qu'est-ce qui pourrait mal tourner ? » — question qui invite chacun à ménager le projet et son auteur —, on déclare : « le projet a échoué, complètement ; racontez pourquoi ». L'astuce n'est pas rhétorique, elle est documentée : se placer *après* l'événement imaginé améliore nettement la capacité à en identifier les causes — le recul prospectif des psychologues de la

décision[5]. Et surtout, le prémortem donne une permission : dans une pièce où l'architecte a déjà tranché, dire « voilà ce qui me semble fragile » est une attaque ; raconter pourquoi le projet mort est mort est un jeu d'équipe. C'est l'outil le moins cher de ce chapitre — une heure, zéro logiciel.

Le quatrième outil est un mot, et il vaut une méthode : *satisficing*. Herbert Simon — le même que celui des sciences de l'artificiel du chapitre « Le mot et la chose » — a montré que les organismes réels, aux capacités bornées, ne cherchent pas l'optimum : ils cherchent le suffisant, et s'arrêtent là[6]. Ce qui ressemble à une paresse est la seule rationalité praticable, et elle répond au défaut structurel des problèmes de conception : le chapitre « Le mot et la chose » a dit, avec Rittel, qu'ils n'ont pas de règle d'arrêt. Un design peut toujours être poli davantage ; c'est pourquoi le seuil de suffisance doit être écrit *dans le cahier des charges*, avant de commencer — au-delà de ce contraste mesuré, au-delà de ce temps de réponse, on s'arrête. Optimiser au-delà du seuil n'est pas de l'exigence, c'est de la procrastination habillée en mathématiques ; et ne pas écrire de seuil du tout, c'est se condamner à trancher chaque arrêt au rapport de force ou à l'épuisement.

Restent les outils dont il faut parler parce qu'ils sont partout, et qu'il faut peser parce que leur aura excède leur preuve. La grille MoSCoW — *Must, Should, Could, Won't* — trie les exigences en 4 urgences ; elle est née chez un praticien d'Oracle en 1994, adoptée par un consortium de méthode agile, enseignée dans toutes les formations — et jamais validée : aucune étude contrôlée à sa naissance, et la seule étude empirique récente qui s'y soit attelée

conclut prudemment qu'elle « pourrait ne pas être aussi efficace qu'on le pensait »[7]. Verdict : un vocabulaire commode pour s'entendre en réunion — c'est déjà quelque chose —, pas une méthode prouvée, et surtout pas un argument. TRIZ, la « théorie de la résolution inventive des problèmes », impressionne davantage : un ingénieur soviétique a dépouillé des dizaines de milliers de brevets pour en extraire 40 principes récurrents d'invention[8]. Le corpus est réel et l'idée — chercher les schémas de solution déjà éprouvés ailleurs — est saine ; mais les revues systématiques constatent que la pratique n'utilise que deux ou trois outils de base sur l'édifice entier, et qu'aucune validation contrôlée n'a jamais suivi les ambitions affichées ; les taux de succès qui circulent dans les plaquettes de formation ne sont adossés à rien[8]. Verdict : une mine d'amorces pour penser par analogie, à piller librement — et une église à ne pas rejoindre, certification comprise.

Ce classement final donne la loi du chapitre. Un outil de conception vaut par le mécanisme qu'il structure — une discussion (Pugh), une mémoire (checklist), une dissidence (prémortem), un arrêt (satisficing) — et par la preuve qui l'étaye, dans cet ordre. Il ne vaut jamais par son acronyme, par son industrie de certification ou par le nombre de ses adeptes : quand la formation payante prospère plus vite que la preuve, ce qu'on achète n'est plus une méthode, c'est une appartenance. Cette partie du livre s'achève donc sur cette mise en garde, qui en annonce une autre : le chapitre « Le culte du cargo » montrera ce que devient une règle — même excellente — quand on la récite sans sa raison. Munis de l'outillage, nous pouvons descendre là

où presque personne ne regarde : sous l'interface, dans la machine. C'est l'objet de la partie « La machine ».

---

1. Pahl, G. & Beitz, W. (2007). *Engineering Design : A Systematic Approach*, 3e éd. (avec J. Feldhusen et K.-H. Grote ; trad. K. Wallace & L. Blessing). Londres : Springer https://doi.org/10.1007/978-1-84628-319-2 (original allemand *Konstruktionslehre*, 1977). Pas de traduction française ; l'ouvrage codifie la tradition dite VDI 2221 : clarification de la tâche, conception de principe, incarnation (*embodiment design*), définition de détail. ↩

2. Ulrich, K. T. & Eppinger, S. D. (2020). *Product Design and Development*, 7e éd. New York : McGraw-Hill (1re éd. 1995 ; 8e éd. 2026 avec M. C. Yang) https://search.worldcat.org/title/Product-design-and-development/oclc/904505863 . Pas de traduction française. ↩

3. Pugh, S. (1981). « Concept Selection : A Method That Works », in Hubka, V. (dir.), *Proceedings of the International Conference on Engineering Design (ICED 81)*, Rome. Zurich : Heurista, p. 497-506 ; développé dans *Total Design : Integrated Methods for Successful Product Engineering*. Wokingham : Addison-Wesley, 1991 https://search.worldcat.org/title/Total-design-:-integrated-methods-for-successful-product-engineering/oclc/878735886 . La méthode originale est qualitative (+/S/– contre un *datum*, « convergence contrôlée » par tours successifs) ; les pondérations

numériques sont des extensions ultérieures, étrangères au texte fondateur. ↩

4. Gawande, A. (2009). *The Checklist Manifesto : How to Get Things Right*. New York : Metropolitan Books https://search.worldcat.org/title/The-checklist-manifesto-:-how-to-get-things-right/oclc/546868192 . Les deux études sous-jacentes : Pronovost, P. *et al.* (2006). « An Intervention to Decrease Catheter-Related Bloodstream Infections in the ICU ». *New England Journal of Medicine*, 355(26), 2725-2732 https://doi.org/10.1056/NEJMoa061115 — le déploiement dans l'État du Michigan (le pilote initial, mené à Johns Hopkins, est Berenholtz, S. M. *et al.*, 2004, *Critical Care Medicine*, 32(10) https://doi.org/10.1097/01.ccm.0000142399.70913.2f) ; et Haynes, A. B. *et al.* (2009). « A Surgical Safety Checklist to Reduce Morbidity and Mortality in a Global Population ». *New England Journal of Medicine*, 360(5), 491-499 https://doi.org/10.1056/NEJMsa0810119 — huit hôpitaux, huit villes, quatre continents. ↩ ↩

5. Klein, G. (2007). « Performing a Project Premortem ». *Harvard Business Review*, 85(9), 18-19 https://hbr.org/2007/09/performing-a-project-premortem . Le fondement expérimental : Mitchell, D. J., Russo, J. E. & Pennington, N. (1989). « Back to the Future : Temporal Perspective in the Explanation of Events ». *Journal of Behavioral Decision Making*, 2, 25-38 https://doi.org/10.1002/bdm.3960020103 — le gain d'environ

30 % dans l'identification des causes est le chiffre que Klein attribue à cette étude. ↩

6. Simon, H. A. (1956). « Rational Choice and the Structure of the Environment ». *Psychological Review*, 63(2), 129-138 https://doi.org/10.1037/h0042769 : « les organismes s'adaptent assez bien pour "satisfaire au suffisant" (*satisfice*) ; ils n'optimisent pas, en général » (p. 129, notre traduction). L'article compagnon : « A Behavioral Model of Rational Choice ». *Quarterly Journal of Economics*, 69(1), 1955, 99-118 https://doi.org/10.2307/1884852 . ↩

7. Clegg, D. & Barker, R. (1994). *CASE Method Fast-Track : A RAD Approach*. Wokingham : Addison-Wesley https://search.worldcat.org/title/CASE-method-fast-track-:-a-RAD-approach/oclc/59952336 . Heuristique conçue chez Oracle, adoptée par le consortium DSDM au début des années 2000, sans validation expérimentale ; l'étude critique récente : « Assessing the Effectiveness of MoSCoW Prioritization in Software Development », *EAI Endorsed Transactions on Internet of Things*, 2024 https://doi.org/10.4108/eetiot.6515 . ↩

8. Altshuller, G. (1999). *The Innovation Algorithm : TRIZ, Systematic Innovation and Technical Creativity* (trad. L. Shulyak & S. Rodman). Worcester, MA : Technical Innovation Center https://search.worldcat.org/title/The-innovation-algorithm-:-TRIZ-systematic-innovation-and-technical-creativity/oclc/42591715 (original russe, 1973 ; en français, du même auteur : *Et soudain apparut l'inventeur*, trad. A. Seredinski, 2006 https://www.decitre.fr/livres/et-

soudain-apparut-l-inventeur-9782952139410.html). Le bilan critique : Ilevbare, I. M., Probert, D. & Phaal, R. (2013). « A Review of TRIZ, and its Benefits and Challenges in Practice ». *Technovation*, 33, 30-37 https://doi.org/10.1016/j.technovation.2012.11.003 ; Chechurin, L. & Borgianni, Y. (2016). « Understanding TRIZ through the Review of Top Cited Publications ». *Computers in Industry*, 82, 119-134 https://doi.org/10.1016/j.compind.2016.06.002 . ↩ ↩

Partie III — La machine

10. Sous l'interface, les dragons

Les cartes anciennes marquaient les régions inexplorées d'une formule : *hic sunt dracones*, ici sont les dragons. C'est très exactement le statut du moteur d'un logiciel dans la plupart des discussions d'interface. Les gens raisonnent sur ce qu'ils voient (les boutons, les panneaux, les curseurs) et sous cette surface, pour eux, il y a des dragons : ils n'y connaissent rien, ils n'y voient rien, donc ils n'en parlent pas. Quiconque ne voit, ne pense et ne se soucie que de l'interface est un idiot dangereux, y compris diplômé en design ou en expérience utilisateur. Incapable de comprendre qu'en dessous vit un organisme, un moteur (ce que le jargon du métier appelle le *backend*, l'arrière-boutique du logiciel ; les deux mots serviront indifféremment ici), il ne voit pas que l'interface n'est que le point de contact avec cet organisme. Cette partie du livre descend chez les dragons, et ce premier chapitre établit son théorème d'entrée : la complexité d'une interface n'est pas une propriété de l'interface.

L'image juste est celle du tableau de bord, parce qu'un tableau de bord ne décore pas : il pilote. Les voitures anciennes portaient un bouton de starter, qui enrichissait le mélange de carburant au démarrage à froid ; il n'était pas là par goût du bouton, mais parce que le moteur à carburateur ne savait pas régler son mélange tout

seul, et que quelqu'un devait donc le faire à sa place — le conducteur était un organe du moteur. Aucun styliste de planche de bord ne pouvait retirer ce bouton : cacher la commande n'aurait pas supprimé la fonction, seulement laissé le mélange sans personne pour le régler. Le starter n'a pas disparu parce qu'on l'a rangé dans un menu ; il a disparu le jour où l'injection électronique a rendu le moteur capable de doser lui-même — la commande est morte à la surface quand la fonction est descendue dans la machine, et la pédale d'embrayage a suivi le même chemin avec la boîte de vitesses automatique. Les paramètres et les contrôles d'une interface logicielle sont exactement cela : non des ornements, mais les commandes d'une machine qui tourne dessous. Si le moteur réclame 6 ou 10 paramètres, on ne peut pas se contenter de les cacher et vivre dans le déni, car quelqu'un devra bien leur donner une valeur. Retirer un élément de l'interface suppose de l'avoir d'abord retiré du moteur, ou d'avoir rendu le moteur capable de s'en passer, ou de le régler lui-même — l'injection, pas le cache-bouton. D'où le théorème, et il faut le prendre au pied de la lettre : **quand l'interface est complexe, c'est déjà trop tard.** Sa complexité, elle l'a héritée, et non créée. L'interface graphique n'est pas parallèle à l'architecture du moteur, elle en est la *terminaison* : le dernier segment, celui qui affleure. On ne résout pas un problème d'interface avec des maquettes ; on le résout en résolvant le problème de moteur dont l'interface est le symptôme.

Ce théorème a un parrainage théorique. Larry Tesler, l'un des pères de l'interaction moderne, a formulé ce qu'on appelle la loi de conservation de la complexité : pour une tâche donnée, il

existe une quantité de complexité irréductible, qui ne peut pas être supprimée, seulement déplacée ; la seule vraie question est de savoir *qui l'absorbe*, de l'utilisateur ou du système[1]. Chaque paramètre que le moteur ne sait pas déterminer lui-même est une part de complexité déléguée à l'utilisateur ; chaque décision que le moteur prend correctement à sa place en est une part absorbée. Absorber à la place de quelqu'un une difficulté qu'il aurait dû porter seul : la formule est d'ingénieur, la chose ne l'est pas. Don Norman en a tiré la leçon d'ensemble : la complexité n'est en soi ni bonne ni mauvaise, c'est la *confusion* qui est mauvaise[2]. Un outil de métier est légitimement complexe, comme le métier l'est ; l'échec de conception, c'est la complexité sans structure, placée au mauvais endroit, qui se vit comme confusion. Ma contribution à cette lignée est causale, et l'expérience me l'a apprise durement : la complexité *coule* dans un sens, du backend vers l'interface. On ne peut donc pas simplifier l'aval sans simplifier l'amont ; toute campagne de simplification qui commence par les pixels finit en cache-misère.

Premier cas : dans Darktable, survoler une vignette de la photothèque déclenchait *18 requêtes* vers la base de données, à chaque survol, pour rafraîchir des métadonnées qui n'avaient pas changé depuis le survol précédent. Le symptôme, côté interface, est une photothèque poussive qui vide les batteries au moindre mouvement du curseur ; la cause n'a rien d'interfacial : le moteur ne suivait pas le cycle de vie de ses données, donc il ne pouvait rien mettre en cache, donc il rafraîchissait tout, tout le temps. C'est le cas du chapitre « La racine et le symptôme », vu depuis l'autre bout. Deuxième cas : l'arbitrage entre la sélection explicite

et la sélection au survol (l'ambiguïté qui écrasait des métadonnées au chapitre « Mesurer sans se mentir ») était « géré » par un fichier d'environ 1 100 lignes, là où une vingtaine suffisent une fois l'ambiguïté tranchée dans le moteur : 1 000 lignes de code pour ne pas prendre une décision de conception et couvrir tous les cas possibles et préférences utilisateur, certains voulant inclure l'image survolée dans leur sélection, et pas les autres. Troisième cas : le système de raccourcis clavier, refondu pour rendre *chaque* action de l'interface configurable, y compris depuis des contrôleurs MIDI et des manettes de jeu. Résultat côté backend : de 1 500 à plus de 4 000 lignes, complexité mesurée en hausse d'un quart ; résultat côté utilisateur : une liste de plusieurs milliers de raccourcis clavier configurables dans laquelle il est devenu difficile de retrouver les 3 raccourcis dont on a réellement besoin. La complexité ajoutée au moteur a coulé jusqu'à l'écran, comme toujours, et la liste d'objets à configurer a atteint une longueur ingérable. Quatrième cas, en sens inverse, car le théorème prédit aussi le remède : Bauhaus, la bibliothèque maison de composants graphiques de Darktable, celle qui dessine les curseurs et les choix déroulants (*combobox*). Symptôme côté interface : des redessins parasites et des lenteurs à dispatcher les changements de valeurs. Le composant attendait la fin du recalcul de l'image pour se redessiner ; l'utilisateur, sans confirmation visuelle que sa valeur était prise en compte, recommençait son geste, ce qui déclenchait un nouveau recalcul sans plus mettre à jour la confirmation visuelle, et la machine se figeait de longues minutes. Côté code: le quatrième fichier le plus complexe de tout le projet, où une simplification que j'avais faite en 2019 avait été effacée par l'inflation de fonctionnalités, dont la prise en charge

MIDI du troisième cas. La réécriture m'a pris 3 semaines à plein temps, pour un principe simple : redessiner immédiatement, ne transmettre la valeur au moteur de calcul qu'à la fin du geste. Complexité cyclomatique réduite d'un tiers, complexité cognitive presque divisée par deux ; et l'interface, sans changer de dessin, est devenue réactive et prévisible. On ne l'a pas simplifiée : on a simplifié ce dont elle hérite.

Un logiciel est un système d'information. Ce qui existe réellement sous le décor, ce sont des données : du texte, de l'image, ses métadonnées, l'historique de retouche, les réglages, les caches. Ces données circulent dans un flux qui les traite, et forme le cycle de vie du logiciel. Concevoir l'architecture, c'est décider comment cette information circule : quelles structures de données existent, quel module en est propriétaire, qui a le droit de les lire et de les écrire, par quelles interfaces les modules se parlent, et comment le cœur du logiciel orchestre l'ensemble — qui est prévenu quand une donnée change, dans quel ordre, et qui tranche en cas de conflit. Voilà l'objet réel du travail de conception logicielle. Et voilà aussi avec quoi l'utilisateur interagit à travers l'interface : un curseur ne « fait » rien, il écrit une valeur dans un paramètre du backend ; un affichage ne « montre » rien, il lit un état du système d'information, qu'il convertit en pixels écran. L'interface graphique n'est qu'un client du système d'information parmi d'autres possibles (la ligne de commande et les scripts en sont d'autres), et sa qualité est bornée par la qualité du flux de données qu'elle consomme. Relisez les 4 cas à cette lumière. Les 18 requêtes par survol : une interface obligée de redemander l'état des données à la base parce que personne ne lui signale ce

qui a changé. Le fichier de 1 100 lignes : une donnée, la sélection, sans propriétaire unique. Les redessins parasites de Bauhaus : un flux mal séquencé entre le geste, l'affichage et le calcul de rendu. Aucun de ces problèmes n'est visible sur une maquette, et tous se voient à l'écran — parce que l'écran ne montre jamais que l'état final du flux d'information.

Comment, alors, un système d'information absorbe-t-il de la complexité sans la propager ? La réponse canonique date de 1972 et n'a pas vieilli : David Parnas a montré que découper un système en modules, c'est décider quelles décisions de conception chaque module *cache* aux autres[3]. Un module bien conçu enferme une décision difficile (un format, un algorithme, un ordre d'exécution) derrière une frontière que le reste du système ne peut pas franchir : la décision peut changer sans que rien d'autre ne bouge, un changement à l'intérieur du module ne peut avoir aucun effet hors de ses frontières, par construction. C'est le même geste que la centralisation des masques du chapitre « Mesurer sans se mentir », une décision, un endroit ; et c'est lui qui fabrique les moteurs dont les interfaces peuvent être simples : chaque décision que l'architecture enferme est un paramètre que l'utilisateur ne verra jamais. L'inverse est tout aussi vrai, et les cas ci-dessus le montrent. Un backend qui ne tranche pas (deux sélections concurrentes, un cycle de vie de données indéterminé) exporte son indécision vers l'interface, où elle devient des options, des incohérences et des accidents. Les architectes disent parfois que l'ossature d'un système finit toujours par se voir ; pour l'interface, c'est une loi : *l'écran est la radiographie de l'architecture.*

L'objection en face de ce chapitre est une orthodoxie professionnelle : concevoir « l'expérience d'abord », c'est-à-dire dessiner l'interface idéale, puis sommer l'implémentation de suivre. Il faut lui concéder sa part de vérité, qui est la nôtre : oui, tout part de l'utilisateur — mais de son *problème* (chapitre « Poser le problème »), pas des pixels. L'erreur commence quand le dessin précède l'architecture du système d'information : le dessin fixe alors des promesses de circulation de données que le backend ne peut pas tenir, et l'on obtient des conceptions inapplicables, en conflit avec ce dont le logiciel a besoin pour fonctionner ; puis, la deadline aidant, l'interface promise se fait greffer sur le moteur existant, et la greffe produit exactement les monstres chiffrés ci-dessus. La vérité inconfortable est qu'il faut des gens qui comprennent les deux étages, le métier de l'utilisateur et l'architecture du moteur, et qu'ils sont rares, donc chers. Mais il n'y a pas d'alternative : les deux étages n'en font qu'un. La planche de bord la plus épurée du monde n'a perdu ses boutons de starter que parce que son moteur a appris à s'en passer.

Si la simplicité d'une interface se gagne dans le moteur, reste à savoir ce que la simplicité vaut, ce qu'elle coûte, et pourquoi tant de logiciels choisissent, case après case, option après option, exactement l'inverse. C'est l'objet du chapitre suivant.

---

1. La « loi de Tesler », formulée vers le début des années 1980, n'a pas de publication primaire ; la source citable est l'entretien de Tesler avec Dan Saffer : Saffer, D. (2006). *Designing for Interaction*. Berkeley : New Riders https://openlibrary.org/works/OL9221137W . L'attribution

populaire « au milieu des années 1980 chez Xerox PARC » est chronologiquement douteuse (Tesler avait quitté le PARC en 1980) ; la loi, elle, est bien de lui. Norman (voir note suivante) la crédite nommément à Tesler, alors chez Apple, au chapitre 2 de son livre. ↩

2. Norman, D. A. (2010). *Living with Complexity*. Cambridge, MA : MIT Press https://openlibrary.org/works/OL15507110W . Verbatim : « Modern technology can be complex, but complexity by itself is neither good nor bad : it is confusion that is bad. » Pas de traduction française à ce jour. ↩

3. Parnas, D. L. (1972). « On the Criteria To Be Used in Decomposing Systems into Modules ». *Communications of the ACM*, 15(12), 1053-1058 https://doi.org/10.1145/361598.361623 — le principe dit de dissimulation de l'information (*information hiding*) : chaque module cache une décision de conception susceptible de changer. ↩

11. L'éloge du simple

Prenez en main un vieux reflex argentique entièrement mécanique. Trois commandes très générales : la bague de mise au point, la bague de diaphragme, le barillet des vitesses. Avec cela, on accomplit ce que les boîtiers numériques font aujourd'hui à grand renfort de dizaines de sous-menus spécialisés. Le numérique a rendu l'appareil plus rapide et plus fiable dans certains contextes, mais il a ajouté une charge d'attention considérable pour piloter, au fond, les mêmes 4 paramètres qu'en 1960 : vitesse, ouverture, (sensibilité), mise au point. La leçon de l'objet tient en une phrase, et elle porte tout ce chapitre : **plus d'options n'est pas plus de contrôle.** Le contrôle vient de commandes générales bien choisies ; les options, passé un seuil, ne s'additionnent plus, elles s'enterrent les unes les autres. Et ce fait ne présume absolument pas de l'efficacité ou de la facilité d'usage de l'outil : nous verrons dans ce chapitre que facile d'usage n'est pas synonyme de simple.

Cacher un arbre dans une forêt est la meilleure méthode pour qu'on ne le trouve plus, et la psychologie expérimentale a mesuré le phénomène dès les années 1950 : le temps de décision croît avec le logarithme du nombre d'alternatives[1]. Ce n'est pas une taxe payée une fois, à l'apprentissage ; elle est payée à chaque usage, par tous les utilisateurs, y compris ceux qui n'ont jamais demandé l'option. La liste de plusieurs milliers de raccourcis configurables du chapitre précédent en est l'illustration : chaque

entrée ajoutée rend les 3 entrées utiles plus difficiles à retrouver. On invoque parfois ici le « paradoxe du choix » et sa célèbre étude des pots de confiture ; la vérité expérimentale est plus prudente, et l'argument n'en a pas besoin[2] : le logarithme de Hick, l'explosion combinatoire et les coûts de maintenance suffisent amplement.

L'explosion combinatoire est le deuxième coût, et il se paie dans le backend. Les options ne s'additionnent pas, elles se multiplient. Dans Darktable, le rendu d'une simple vignette d'image devait composer avec 5 vues et 7 variantes d'affichage, plus la bande de défilement : 36 combinaisons, multipliées par 3 thèmes graphiques, soit 108 jeux d'instructions de style CSS pour habiller complètement l'interface. Lors d'une refonte graphique majeure, plusieurs de ces jeux furent tout simplement oubliés ; comment aurait-il pu en être autrement ? L'équipe de Darktable s'est implicitement engagée — sans le savoir — à maintenir et à tester 108 combinaisons de styles sur l'ensemble des widgets graphiques de la table lumineuse, alors que tout cela ne semblait qu'un innocent effort pour accommoder tout le monde avec chacun sa variante. Un simple calcul aurait suffit à annuler le projet au motif de l'ampleur démesurée de la tâche (pour un projet qui, on se le rappelle, ne souhaitait pas fournir de paquets logiciels pré-compilés au motif de la charge de travail que cela représente), si le coût de maintenance avait fait partie de la checklist du chapitre « S'outiller sans se soummettre », et si l'analyse combinatoire de la surface d'options à maintenir faisait partie des compétences de l'équipe. Aucun humain ne teste 108 combinaisons de styles sur l'intégralité de l'interface, donc

l'espace d'états impossible à tester produit ce qu'il produit toujours : des bugs irreproductibles, qui dépendent d'une configuration que le rapporteur ne pense même pas à mentionner. Et il existe un troisième coût, plus pervers : l'option qui permet à l'utilisateur de se nuire. Le même logiciel s'est doté d'un outil de disposition des modules entièrement configurable, sans guide de bonnes pratiques, qui enregistrait la disposition sous le nom *traduit* des modules : changer la langue de l'interface suffisait à perdre ses réglages. Offrir cette latitude n'est pas un service ; c'est laisser l'utilisateur décider comment se faire mal, et lui en laisser la responsabilité. Reste que la lui refuser peut être pire encore, et l'objection porte un nom — le paternalisme : décider à la place de l'autre, pour son bien. Le chapitre « Ouvrir sans tenir le pinceau » tracera la ligne, et le critère y sera le même d'un bout à l'autre de ce livre : protéger quelqu'un, c'est le rendre capable de décider, jamais décider pour lui, mais en rendant l'option dangereuse tout de même un peu plus coûteuse que l'autre pour s'assurer que c'est bien ce qu'il souhaite. C'est le principe de la commande de forçage.

Un outil n'est pas la somme de ses fonctionnalités. Le couteau suisse en est la preuve par l'absurde : bardé de fonctions, et pourtant aucun artisan n'a remplacé sa caisse à outils par un couteau suisse, parce qu'il est peu ergonomique, fragile, et qu'on ne serre pas une vis à un couple suffisant avec un manche plat de 5 centimètres. Utile en camping ; hors de question pour le travail. L'atelier fournit l'image complémentaire : quand les établis, les étagères et les tables sont couverts d'outils ultra-spécialisés, la productivité marginale de chaque outil supplémentaire devient

négative, parce que le temps que fait gagner l'outil se perd à le chercher parmi les autres, et la loi de Murphy veut qu'il soit toujours au fond du tiroir ou de la boîte. Mieux vaut un nombre d'outils maîtrisé, généraux et complémentaires, et des tables réservées au travail. Passé la complétude fonctionnelle, continuer d'ajouter transforme l'outil en jouet : un objet plein de fonctions excitantes qui ne rend pas plus productif, mais fait perdre du temps à chercher ce dont on a besoin parmi ce qui encombre. Un outil se développe jusqu'à ce qu'il remplisse son objet ; ensuite, développer encore n'est plus améliorer. Le développement abîme la stabilité et engendre de la maintenance : c'est littéralement du travail qui crée du travail. Un projet mûr doit ralentir, et ce n'est pas un aveu de faiblesse, c'est un signe d'achèvement.

Cette économie a des règles pratiques. Chaque fonctionnalité s'évalue contre son coût de maintenance à vie, pas contre son attrait au moment de l'écrire ; le principe a un nom chez les praticiens, YAGNI, *you aren't gonna need it* : on implémente quand le besoin est là, jamais parce qu'il pourrait venir[3]. La priorité se règle sur les données d'usage du chapitre « Compter les absents », et l'exemple négatif est déjà dans ce livre : doubler la complexité du système de raccourcis pour servir les contrôleurs MIDI, possédés par un utilisateur sur 10 et utilisés par deux sur 100, est un contresens de priorités. Enfin, la règle d'hygiène que j'applique depuis le premier jour de mon fork, où 22 000 lignes ont été retirées d'un coup : ce qui fonctionne à moitié est supprimé en entier. Une fonctionnalité à moitié fonctionnelle coûte le prix d'une fonctionnalité entière en maintenance, rapporte une fraction du service, et fabrique de la

déception en continu chez l'utilisateur. Retirer du code et des options : la robustesse est à ce prix.

L'éloge du simple n'est pas l'éloge du simpliste. Mais le vrai malentendu est plus profond : quand on réclame de la *simplicité*, on veut presque toujours dire de la *facilité*, et l'on croit que les deux vont ensemble. Elles s'opposent. Un violon est l'objet le plus simple du monde : 4 cordes, un archet, 4 chevilles, une forme inchangée depuis 4 siècles. Il n'a rien de facile ; il faut des années pour en tirer un son juste. Le simple, c'est l'outil dépouillé et général, celui auquel la main doit s'adapter et s'entraîner. Sa difficulté appartient au métier, pas à l'objet.

Le facile est l'inverse, et le mécanisme est contre-intuitif : on ne rend pas un outil facile en le simplifiant, on le rend facile en le *spécialisant*. Dans le monde des machines mécaniques, spécialiser signifie retirer des usages possibles et restreindre le champ d'usage, pour optimiser la machine et son interface à un usage prévu (ou à quelques usages). Dans le monde flexible des machines informatiques, la facilité s'achète en ajoutant des modes, des préréglages, des fonctions taillées d'avance pour des scénarios d'usage prévus — de sorte que l'utilisateur n'ait rien à apprendre, l'appareil ayant déjà décidé pour lui. Chaque scénario prévu est une machine spécialisée de plus. Et comme un logiciel empile les machines spécialisées sans limite physique, on les multiplie pour couvrir toute la gamme des usages, jusqu'à ce que l'interface en soit saturée et devienne illisible. La facilité poursuivie produit donc, très exactement, l'encombrement que la première moitié de ce chapitre a mesuré. C'est pourquoi la demande de « simplicité » débouche si souvent sur son contraire :

ce qu'on réclame sous ce nom, c'est de la facilité, et la facilité se paie en anti-simplicité.

Simplifier un outil de travail, ce n'est donc pas le rendre facile. C'est réduire le nombre d'étapes et la charge cognitive *pour la même tâche accomplie*, sans jamais dispenser du métier. La difficulté doit rester au métier ; elle ne doit jamais passer à l'outil. Et l'inverse du même piège existe : les applications à l'apparence la plus « simple » obtiennent souvent cette façade en dégradant ce qui ne se voit pas, à commencer par leurs modèles de couleur. Tout a un coût ; la loi de Tesler du chapitre précédent ne prend pas de vacances. Le chapitre « L'instrument et la machine » développera cette opposition — l'instrument contre la machine — pour les outils de création.

L'objection en face a la meilleure réputation qui soit : la flexibilité comme liberté. Qui suis-je pour décider à la place de l'utilisateur ? Que chacun configure son outil selon son flux de travail ; les options respectent la diversité des pratiques. Cette objection tire sa force de biais bien documentés. Retirer une fonction est vécu comme une *perte*, et la psychologie économique a établi qu'une perte pèse plus lourd qu'un gain équivalent : c'est l'aversion à la perte, doublée du biais de statu quo qui fait préférer l'état présent au seul motif qu'il est présent[4]. S'y ajoute le poids de l'effort investi : plus on a passé de temps à configurer un outil, plus on le surévalue — l'« effet IKEA », où le travail fourni engendre l'attachement, indépendamment du résultat[5]. L'utilisateur assimile alors les options qu'il a accumulées à son flux de travail, et retirer une fonction se vit comme l'amputation d'une pratique — d'où la peur, récurrente dans le libre, qu'une simplification

« casse le workflow de quelqu'un ». Mais une préférence n'est pas un flux de travail, et l'empilement d'options n'en construit aucun. Le marketing l'a mesuré sous le nom de *feature fatigue* : avant l'achat, le client choisit le produit le plus riche en fonctions ; à l'usage, cette abondance dégrade sa satisfaction, la capacité acquise se payant en complexité subie[6]. L'option n'oriente personne, et surtout pas le débutant, qui a besoin d'être guidé, pas mis devant un tableau de bord. Trois réponses, donc. D'abord, le droit de chacun s'arrête où commence la taxe de tous : le logarithme de Hick est payé par tous les utilisateurs, à chaque usage, y compris par ceux qui ne voulaient rien configurer ; l'option de l'un encombre l'écran de l'autre. Ensuite, le palmarès réel des options ajoutées au nom de la liberté est consultable : 108 jeux de styles dont on oublie de tester la moitié, des réglages perdus au changement de langue, des milliers d'entrées dans le fichier de configuration. Enfin et surtout, la confusion porte sur le mot : la vraie flexibilité est la *généralité*, pas la configurabilité. L'outil générique — la bague de diaphragme, l'égaliseur de tons — sert des usages que son concepteur n'a pas prévus, précisément parce qu'il expose un petit nombre de degrés de liberté fondamentaux au lieu de multiplier les cas particuliers préconfigurés. La bonne réponse à la diversité des pratiques est un outil général qui se plie à tout, pas un outil configurable qui se déplie en 108 états. L'option garde une légitimité résiduelle, et le chapitre « Poser le problème » a donné le critère : elle se justifie quand elle sert des *besoins* incompatibles — l'accessibilité, les tailles d'écran, le clavier seul et la souris seule — jamais quand elle arbitre des préférences.

Faire moins, mieux : le programme tient en 3 mots, et il a un ennemi structurel — le code lui-même, qui s'accumule plus volontiers qu'il ne se retire, et dont chaque ligne est une dette dont on paie les intérêts. C'est l'objet du chapitre suivant.

---

1. Hick, W. E. (1952). « On the rate of gain of information ». *Quarterly Journal of Experimental Psychology*, 4(1), 11-26 https://doi.org/10.1080/17470215208416600 ; Hyman, R. (1953). « Stimulus information as a determinant of reaction time ». *Journal of Experimental Psychology*, 45(3), 188-196 https://doi.org/10.1037/h0056940. Le temps de réaction de choix croît en $\log_2(n)$ du nombre d'alternatives (loi de Hick-Hyman). ↩

2. L'étude des confitures : Iyengar, S. S. & Lepper, M. R. (2000). « When Choice is Demotivating : Can One Desire Too Much of a Good Thing ? ». *Journal of Personality and Social Psychology*, 79(6), 995-1006 https://doi.org/10.1037/0022-3514.79.6.995 ; popularisée par Schwartz, B. (2004). *The Paradox of Choice*. New York : Ecco https://openlibrary.org/books/OL26343625M/The_Paradox_of_Choice. Mais la méta-analyse de Scheibehenne, B., Greifeneder, R. & Todd, P. M. (2010), « Can There Ever Be Too Many Options ? », *Journal of Consumer Research*, 37(3), 409-425 https://doi.org/10.1086/651235, trouve un effet moyen proche de zéro : la surcharge de choix n'apparaît que sous conditions. L'argument du chapitre ne repose donc pas sur elle. ↩

3. Beck, K. (1999). *Extreme Programming Explained : Embrace Change*. Reading, MA : Addison-Wesley https://books.google.com/books/about/Extreme_Programming_Explained.html?id=G8EL4H4vf7UC — la règle « You Aren't Gonna Need It » : n'implémenter que ce dont le besoin est avéré, la fonctionnalité anticipée étant presque toujours mal anticipée. ↩

4. Kahneman, D., Knetsch, J. L. & Thaler, R. H. (1991). « Anomalies : The Endowment Effect, Loss Aversion, and Status Quo Bias ». *Journal of Economic Perspectives*, 5(1), 193-206 https://doi.org/10.1257/jep.5.1.193 — on surévalue ce qu'on possède déjà (effet de dotation) ; une perte pèse davantage qu'un gain égal (aversion à la perte) ; on préfère l'état en place pour lui-même (biais de statu quo). ↩

5. Norton, M. I., Mochon, D. & Ariely, D. (2012). « The IKEA Effect : When Labor Leads to Love ». *Journal of Consumer Psychology*, 22(3), 453-460 https://doi.org/10.1016/j.jcps.2011.08.002 — l'effort investi dans un objet augmente la valeur qu'on lui prête, indépendamment de sa qualité réelle. ↩

6. Thompson, D. V., Hamilton, R. W. & Rust, R. T. (2005). « Feature Fatigue : When Product Capabilities Become Too Much of a Good Thing ». *Journal of Marketing Research*, 42(4), 431-442 https://doi.org/10.1509/jmkr.2005.42.4.431 — avant l'achat, le client privilégie le produit le plus riche en fonctions ; à l'usage, cette abondance dégrade la satisfaction

et l'utilisabilité, le nombre de capacités se retournant contre le produit. ↩

12. Le code est un passif

Il y a une erreur de comptabilité au cœur de la culture logicielle : inscrire le code à l'actif du bilan. On mesure le travail au volume de code ajouté, on célèbre les gros apports, on prend l'agitation des dépôts pour de la santé. C'est le sens inverse du vrai. Le code s'inscrit au passif : chaque ligne écrite est un engagement de dépenses futures, car elle devra être lue, comprise, testée, adaptée et réparée par des humains, pendant toute la vie du logiciel. La métaphore comptable n'est pas de moi : c'est la dette technique de Ward Cunningham, forgée pour expliquer qu'un raccourci pris dans le code porte intérêts jusqu'à son remboursement[1]. Et l'intérêt court même sans raccourci : les lois de Lehman, établies empiriquement dès les années 1970, disent qu'un programme utilisé doit évoluer sans cesse ou mourir, et que sa complexité croît mécaniquement à mesure qu'il évolue, sauf travail explicite pour la réduire[2]. Le code est un jardin : chaque saison apporte son lot de corvées. Créer une voiture n'est pas empiler aveuglément de l'acier sur 4 roues ; à un moment, la voiture doit rouler. La réussite se mesure aux problèmes d'utilisateurs résolus, jamais au tonnage de code produit.

De cette comptabilité découle la définition du bon code, et elle n'a rien d'esthétique : le bon code est le code facile à maintenir. On n'écrit pas du code pour les ordinateurs ; le compilateur avale n'importe quoi. On écrit du code pour les humains qui devront le lire, le comprendre et le réparer, et tout imbécile peut écrire du

code qu'une machine comprend — écrire du code qu'un humain comprend est le métier[3]. Kernighan et Plauger en ont donné le corollaire le plus utile : déboguer son code est deux fois plus difficile que l'écrire ; si vous écrivez le code le plus astucieux dont vous êtes capable, vous êtes par définition trop peu astucieux pour le déboguer[4]. L'ennui est une qualité : le design ennuyeux, prévisible, sans malice, est le meilleur design. La maintenance, elle, est le contraire du prestige : invisible, jamais applaudie, reléguée en fin de notes de version, pendant que les utilisateurs jugent la santé d'un projet à son agitation. On ne gagne rien à maintenir ; on perd tout à ne pas maintenir ; on paie dans les deux cas. Ce verbe est d'ailleurs bien mal logé du côté de la corvée : les philosophes qui ont pensé le soin en ont fait le premier mot de leur définition — maintenir, perpétuer et réparer un monde pour qu'on puisse y vivre. C'est exactement le travail que ce chapitre décrit, et exactement celui que le milieu méprise. La sagesse consiste donc à rendre la maintenance d'abord possible, ensuite facile, et 3 règles y suffisent presque :

1. Écrire le moins de code possible : chercher un bug dans 300 lignes est un autre travail que de le chercher dans 3 000.
2. Écrire le code le plus simple possible : peu d'étapes, peu d'hypothèses, peu de cas particuliers, c'est-à-dire, en pratique, éviter les options, les variantes et les embranchements dont le chapitre précédent a fait le procès.
3. Écrire du code refermé sur lui-même (autonome) : des modules isolés qui communiquent par le cœur du système, pour que ce qui change à l'intérieur d'un module ne casse

rien au-dehors — c'est Parnas, encore, et c'est la seule des 3 règles qui survive à la croissance du logiciel.

Une quatrième règle enveloppe les 3 autres : réfléchir avant d'écrire. La quantité de code produite est inversement proportionnelle à la quantité de réflexion investie, et l'on n'étend jamais un programme qu'on n'a pas d'abord su simplifier.

Réfléchir avant d'écrire a une forme précise en ingénierie : trouver le bon objet mathématique. En 2011, le fondateur de Darktable avait réglé l'ordre des modules dans le pipeline de traitement par la méthode canonique : les développeurs déclarent des contraintes locales sur l'ordonnancement des modules (« A avant C », « B avant A »), un script en fait un graphe orienté, et un tri topologique, algorithme réglé depuis les années 1960, en déduit l'ordre global par résolution de graphe. Propre, minimal, correct. En 2019, il a fallu permettre à l'utilisateur de réordonner les modules à l'exécution. La fonctionnalité fut implémentée en donnant à chaque module une priorité numérique, un nombre à virgule flottante, l'ordre du pipeline étant celui des priorités croissantes. Pour insérer un module entre deux autres, on lui attribue un nombre intermédiaire : entre 1,0 et 2,0, on place 1,5 ; entre 1,0 et 1,5, on place 1,25, et ainsi de suite (bissection). Le procédé paraît malin et il est faux deux fois. D'abord les insertions répétées resserrent les nombres jusqu'à épuiser la précision en virgule flottante de la machine, qui est finie, et deux priorités deviennent indistinguables et créèent des erreurs insolubles. Ensuite et surtout, il encode une *position absolue* là où le problème est fait de *relations* : ce qu'on sait d'un module, c'est qu'il doit venir avant ou après tel autre, pas qu'il habite au

numéro 1,25. Le tout fut greffé sur un code d'historiques qu'on n'avait ni refactorisé ni simplifié au préalable. Résultat : un backend si fragile que le mainteneur qui avait accepté le changement dut le réécrire entièrement l'année suivante avec une structure qui dit ce qu'elle veut dire, la liste chaînée, où chaque module pointe vers le précédent et le suivant. Quant au tri topologique, seul capable de fusionner deux pipelines dont les modules diffèrent, le script pré-compilation qui le faisait avait été supprimé en 2019 sans jamais être porté dans le logiciel pour un usage dynamique : la capacité mathématique du système avait régressé. La morale tient en une ligne : l'heuristique bricolée est une dette, l'objet mathématique juste est un remboursement. Un graphe, un tri, une liste chaînée ne sont pas des raffinements de théoricien ; ce sont les formes les plus courtes et les plus sûres de représenter le problème pour le résoudre correctement, et celles qui font écrire le moins de code. Reste à savoir *quand* réécrire plutôt que réparer, car la réécriture est elle-même une dette potentielle. Le critère n'est pas l'âge du code ni son déplaisir : c'est le changement d'exigence. On répare tant que le besoin est le même et que le défaut est local ; on réécrit quand le besoin a changé au point que la structure ne peut plus le porter — et alors on réécrit *par incréments*, en enveloppant l'ancien et en détournant les appels un à un vers le neuf, jamais par une refonte d'un bloc qui remplace tout d'un coup. Refaire à neuf l'ensemble d'un logiciel qui marche est la faute stratégique classique, celle qui a coulé des produits entiers : on jette avec le code tout le savoir accumulé dans ses recoins, pour réintroduire les bugs qu'il avait fini par corriger[5]. Réparer une fondation éprouvée vaut presque toujours mieux qu'un prototype neuf. Ceci se résume

comme suit : utiliser les bonnes structures de données, et utiliser les bons algorithmes par-dessus. À la date de rédaction, ces bons algorithmes sont la plupart du temps connus et publiés depuis plus de 50 ans, et il est vain de les réinventer, ce qui implique de rechercher l'état de l'art avant de se lancer : encore une tâche « administrative » avant de pouvoir enfin construire quelque chose, ce qui ne plaide pas en sa faveur dans un contexte où l'une des motivations principales du développeur open-source est le plaisir de construire quelque chose en un week-end.

Reste la question des dépendances, car on n'écrit jamais tout soi-même, et un mantra gouverne ce choix : « ne réinventez pas la roue ». La formule est ignorante. La roue est réinventée en permanence, et précisément pour la performance : la roue de charrette en bois, la roue à rayons d'acier, la jante double-paroi en aluminium, la roue lenticulaire en carbone des contre-la-montre n'ont en commun que d'être rondes. Chaque réinvention servait un besoin que la roue précédente ne servait pas. En logiciel, « ne pas réinventer » signifie dépendre d'une bibliothèque tierce, et la dépendance n'est pas du code gratuit : c'est un transfert de passif assorti d'une nouvelle surface de contrat. Les bibliothèques changent leurs interfaces, et le code qui marche aujourd'hui casse demain, à l'heure choisie par quelqu'un d'autre. Certaines sont des bibliothèques de Schrödinger : l'accès aux appareils photo qui fonctionnait il y a deux ans ne fonctionne plus aujourd'hui, rien dans les notes de version ne le mentionne, et c'est le développeur de l'application que l'utilisateur prendra pour un incapable, pas celui de la bibliothèque partagée. Les distributions compilent avec leurs propres options : un format de fichier raw désactivé par le

packageur m'a coûté deux heures de diagnostic, à moi développeur ; l'utilisateur ordinaire n'avait aucune chance. Et en calcul haute performance, les bibliothèques génériques rechargent l'image pixel par pixel à chaque étape, là où un code maison fusionne les boucles et vectorise, en combinant plusieurs opérations dans un seul accès mémoire.

Les agents de programmation par intelligence artificielle, basés sur des LLM, changent d'ailleurs la donne à ce niveau là : il est désormais facile, rapide et fiable de laisser un LLM extraire d'une bibliothèque partagée les 2 ou 3 fonctions dont on a besoin, en les adaptant directement à nos besoins propres et à nos structures de données, et en permettant éventuellement de les optimiser à notre cas d'utilisation. On gagne alors sur les 3 tableaux : en réduisant le caractère fastidieux de la ré-implémentation d'algorithmes classiques, en réduisant la dépendance aux bugs présents dans du code tierce-partie, et en se mettant à l'abri des futures changements d'API dudit code.

Ma définition du bonheur tient en une fraction : le bonheur est l'inverse du nombre de gens dont on dépend. La réconciliation avec le chapitre « Sous l'interface, les dragons » est nette : on réinvente le moteur quand on a des exigences que la bibliothèque générique ne sert pas et qu'on souhaite maîtriser la surface de panne du code, mais on ne réinvente jamais les conventions d'interaction. e double-clic a été inventé il y a 40 ans et les utilisateurs n'ont pas à réapprendre le monde pour vos beaux yeux. Quant à résoudre l'enfer des dépendances en livrant chaque application avec sa copie privée de toutes ses bibliothèques (paquets logiciels Snap, Flatpak), dans des conteneurs qui

tournent en tâche de fond, c'est soigner l'obésité par l'obésité. Sous la moquerie il y a une valeur qu'il faut énoncer : les cycles processeur gaspillés, les dépendances qui gonflent et la consommation à vide ne sont pas des détails d'intendance, ce sont des fautes d'ingénierie. L'efficience est l'objet même du métier — faire beaucoup avec peu —, et un logiciel qui dilapide l'énergie de la machine dilapide celle du réseau qui l'alimente. La frugalité n'est pas une austérité imposée à l'utilisateur ; c'est la marque du travail bien fait, et son contraire, celle d'un travail qui a cessé de compter ce qu'il coûte.

L'objection qui coiffe le tout est la culture de la rapidité : livrer vite de la valeur, assembler des bibliothèques, ne pas s'attarder — le code est devenu bon marché, l'IA le rend presque gratuit. La réponse est dans les dossiers de ce chapitre. Le réordonnancement des modules par priorités numériques était l'option rapide : livré vite, réécrit entièrement l'année suivante — le temps gagné a été largement perdu à la réparation des négligences qui avaient permi le gain de temps en premier lieu. Assembler des bibliothèques qu'on n'audite pas est rapide aussi, jusqu'au jour où l'une d'elles cesse de fonctionner et c'est vous que l'utilisateur prend pour l'incapable. La rapidité confond le coût d'écriture, qui n'a jamais été le problème, avec le coût de possession de code, qui l'a toujours été. Il faut itérer vite, le chapitre « Décider, puis tenir » l'a défendu ; mais itérer vite sur les solutions n'autorise pas à accumuler vite du passif, car la vitesse des itérations suivantes s'achète avec la parcimonie des précédentes. Le seul actif d'un logiciel, c'est le problème résolu et

l'architecture qui le résout lisiblement. Tout le reste, chaque ligne, est au passif, et les intérêts courent.

La bonne machine est disciplinée : simple en surface si elle est simple en profondeur, parcimonieuse dans son code, souveraine sur ses dépendances. Les 3 règles de ce chapitre, comme tout principe qui a fait ses preuves, ont un revers : on peut les réciter sans les comprendre, et une règle récitée sans être comprise finit toujours par produire l'inverse de ce qu'elle promettait. C'est l'objet du chapitre suivant, avant de pouvoir enfin dire à quoi cette machine doit ressembler pour celui qui s'en sert.

---

1. Cunningham, W. (1992). « The WyCash Portfolio Management System ». Rapport d'expérience OOPSLA '92, *ACM OOPS Messenger*, 4(2), 1993 https://doi.org/10.1145/157710.157715 — première formulation de la métaphore de la dette : livrer du code immature, c'est emprunter ; « chaque minute passée sur du code pas-tout-à-fait-juste compte comme un intérêt sur cette dette ». ↩

2. Lehman, M. M. (1980). « Programs, Life Cycles, and Laws of Software Evolution ». *Proceedings of the IEEE*, 68(9), 1060-1076 https://doi.org/10.1109/proc.1980.11805 — notamment la première loi (changement continu : un programme utilisé doit s'adapter ou devenir progressivement moins utile) et la deuxième (complexité croissante : sauf travail dédié, la structure se dégrade à mesure que le programme évolue). ↩

3. Fowler, M. (1999). *Refactoring : Improving the Design of Existing Code*. Reading, MA : Addison-Wesley https://openlibrary.org/books/OL7407595M, p. 15 : « Any fool can write code that a computer can understand. Good programmers write code that humans can understand. » ↩

4. Kernighan, B. W. & Plauger, P. J. (1978). *The Elements of Programming Style*, 2e éd. New York : McGraw-Hill https://openlibrary.org/books/OL4716911M, p. 10 : « Everyone knows that debugging is twice as hard as writing a program in the first place. So if you're as clever as you can be when you write it, how will you ever debug it ? » ↩

5. Spolsky, J. (2000). « Things You Should Never Do, Part I ». Joel on Software, 6 avril 2000 https://www.joelonsoftware.com/2000/04/06/things-you-should-never-do-part-i/ — la réécriture intégrale comme « pire erreur stratégique » (l'exemple de Netscape). La méthode incrémentale : Fowler, M. (2004). « StranglerFigApplication » (martinfowler.com) https://martinfowler.com/bliki/StranglerFigApplication.html — remplacer une application héritée en l'étranglant par les bords, fonction par fonction, jamais d'un bloc. ↩

13. Le culte du cargo

Pendant la guerre du Pacifique, des îles mélanésiennes ont vu atterrir des avions chargés de marchandises. Après la guerre, les avions ont cessé de venir ; alors certains insulaires ont construit des pistes, allumé des feux tout du long, bâti une cabane de contrôleur avec un homme dedans, deux morceaux de bois sur les oreilles en guise de casque et des tiges de bambou en guise d'antennes. La forme était parfaite, mais aucun avion n'a atterri. Richard Feynman a fait de cette image le nom d'une pathologie de la science : des travaux qui suivent tous les rites apparents de la recherche — protocoles, chiffres, publications — mais auxquels il manque l'essentiel, l'honnêteté qui cherche à se réfuter soi-même[1]. Le logiciel a repris l'image dès les années 1990 pour nommer sa propre version du mal : du code rituel, des structures qu'on recopie parce qu'elles ont toujours été là, des méthodes qu'on singe sans comprendre ce qu'elles accomplissent[2]. Ce chapitre applique le diagnostic aux bonnes pratiques elles-mêmes. Car le chapitre précédent a donné des règles — le moins de code, le plus simple, le plus clos — et ce livre serait incomplet s'il ne montrait pas comment chacune de ces règles, récitée sans sa raison, produit exactement ce qu'elle devait empêcher. Une bonne pratique est la forme comprimée d'une douleur passée ; qui applique la forme sans connaître le mécanisme construit des antennes en bambou.

Commençons par la plus vénérée : *don't repeat yourself*, DRY, ne te répète pas. La règle est saine et son texte fondateur précis : ce qui ne doit exister qu'en un seul exemplaire, c'est chaque élément de *connaissance* du système — une formule, une règle métier, un format — pas chaque suite de lignes qui se ressemblent[3]. Les auteurs eux-mêmes ont passé l'édition du vingtième anniversaire à se plaindre du contresens : « beaucoup ont cru que DRY parlait du code — que DRY voulait dire ne copiez-collez pas de lignes de source »[3]. Le rituel, lui, traque la ressemblance visuelle : deux blocs qui se ressemblent, vite, une abstraction commune. Or deux codes semblables ne portent pas forcément la même connaissance — ils peuvent se ressembler par accident, aujourd'hui, et diverger demain. L'abstraction forcée les soude ; chaque cas nouveau y ajoute alors un paramètre, un embranchement, un drapeau, jusqu'à ce que la fonction « commune » soit un labyrinthe où plus personne ne comprend aucun des cas. Une praticienne respectée du courant orienté-objet a donné à cette impasse son nom durable — la mauvaise abstraction — et sa règle de sortie, qui est un sacrilège apparent : « préférez la duplication à la mauvaise abstraction », démontez la fonction commune, réintroduisez les copies, et attendez de comprendre[4]. Le garde-fou existait d'ailleurs dès l'origine du mouvement refactoring : la règle de trois — on tolère la deuxième occurrence, on n'abstrait qu'à la troisième, quand la vraie structure commune est devenue visible[4].

J'ai payé pour apprendre que ce sacrilège est la sagesse même, et la preuve vient de mon propre chantier. Dans Darktable, la grille de vignettes — la table lumineuse — et la bande de défilement

horizontale — le *filmstrip* — affichaient toutes deux des miniatures d'images. Deux affichages semblables : le réflexe DRY s'est déclenché, et un seul et même widget a servi les deux vues, *re-parenté* de l'une à l'autre selon le contexte. Sur le papier, zéro duplication, c'est élégant. En pratique, l'état du widget — position de défilement, initialisation des miniatures, libération de la mémoire, durée de vie des événements — dépendait désormais de la vue où il avait *vécu en dernier*. Défilement qui saute, vignettes détruites au mauvais moment, ramasse-miettes imprévisible ; et le re-parentage était si coûteux que chaque bascule entre table lumineuse et chambre noire gelait l'interface près d'une seconde. Tout cela fut corrigé rustine après rustine, c'est-à-dire jamais : chaque correction ajoutait un cas particulier au monstre. La vraie correction, dans Ansel, a été d'arrêter d'être malin : deux tables de vignettes séparées, une par vue. Un peu plus de structure explicite, beaucoup moins d'états accidentels — et la bascule entre les vues rendue instantanée. La duplication qu'on avait évitée était de 20 lignes ; la non-duplication a coûté des années de bugs. DRY n'avait pas menti : c'est sa récitation qui mentait, car les deux vues ne partageaient pas une connaissance, seulement une apparence.

Le même procès vaut pour le *code propre*, et il se mène au cordeau, car le chapitre précédent défend la lisibilité comme l'économie fondamentale du code. Le canon du mouvement — fonctions minuscules, indirection systématique, polymorphisme partout — a été érigé en morale professionnelle : du code « propre » ou « sale », des développeurs propres ou sales[5]. Or ces règles ont un domaine de validité, et il a des frontières. Un

praticien du calcul haute performance a refait l'exemple du canon — les aires de formes géométriques, chapitre 6 du livre fondateur — en retirant une à une les vertus prescrites :

- sans polymorphisme, 1,5 fois plus vite ;
- en tables simples, 10 fois ;
- vectorisé, 20 fois et plus[5].

C'est une critique de praticien, non revue par les pairs, et elle ne condamne pas la lisibilité : elle borne son royaume. Dans le code de liaison — interfaces, configuration, orchestration —, la hiérarchie d'objets et la petite fonction règnent à bon droit : ce code-là change souvent et se lit plus qu'il ne s'exécute. Dans une boucle de pixels traversée 60 fois par seconde sur 40 mégapixels, chaque indirection se paie en millions d'exemplaires ; là, le tableau plat et la boucle fusionnée ne sont pas de la saleté, ils sont la propreté du domaine. Réciter « petites fonctions, beaucoup d'objets » au milieu d'un pipeline d'image, c'est appliquer l'étiquette du salon dans une salle des machines — la forme du propre, à l'endroit exact où elle salit. Et la même leçon vaut à l'envers pour l'optimisation manuelle : le compilateur optimise d'autant mieux que la logique est simple, et le code « astucieusement » tordu pour aller vite finit régulièrement plus lent une fois compilé. Dans les deux directions, la règle est la même : c'est le profil mesuré qui décide, pas la doctrine.

Les *design patterns* — ces 23 schémas de conception catalogués par la « bande des quatre » et récités depuis comme un bréviaire — offrent le cas le plus instructif, parce que la critique vient de l'intérieur de l'élite du métier[6]. Le catalogue est utile comme

vocabulaire : nommer « observateur » ou « fabrique » ce qu'on allait décrire en trois phrases. Mais Peter Norvig a montré que 16 des 23 patrons deviennent invisibles ou triviaux dans des langages de programmation plus expressifs — ils ne décrivent pas des lois de la conception, mais des *manques du langage* que le programmeur comble à la main par sa propre discipline[6]. Paul Graham en a tiré la conclusion brutale : voir des patterns se répéter dans son code, c'est voir la preuve qu'on écrit à la main, encore et encore, ce que la machine devrait générer — « quand je vois des patterns dans mes programmes, j'y vois un signe d'ennuis »[6]. Le culte inverse la lecture : là où le maître voit un symptôme, le rituel voit une vertu, et l'on juge un code d'autant plus professionnel qu'il empile les patrons. Appliquer un pattern n'est ni bien ni mal : c'est un coût — une couche d'indirection — qui s'échange contre un besoin réel de découplage. Sans le besoin, il ne reste que le coût, porté avec fierté.

Reste la pratique la plus institutionnalisée : la revue de code. Sa légende dit qu'elle attrape les bugs, et la légende a été mesurée. Chez Microsoft, les défauts sont la première motivation déclarée des relecteurs — et un huitième seulement des commentaires réels, loin derrière les améliorations de forme, la compréhension et la conversation technique[7] ; chez Google, l'histoire interne est assumée : la revue a été instituée non pour trouver des bugs, mais pour forcer le code à rester lisible et maintenable par d'autres, et la latence médiane d'une revue y est inférieure à 4 heures — un régime de transmission, pas d'inspection[7]. Autrement dit, la revue de code marche, mais pas pour la raison que le rituel invoque : elle est le mécanisme par lequel une équipe se transmet

la compréhension et défend la lisibilité — la définition même du bon code au chapitre précédent. La version cargo de la revue existe pourtant, et elle est double : l'approbation-tampon, où l'on vise sans lire pour nourrir la métrique ; et son symétrique, le test-quitus que le chapitre « L'outil transparent » racontera, où le feu vert d'une machine dispense de chercher ce qu'elle a vu. Dans les deux cas, la forme de la vérification remplace la vérification.

Et j'ai pu poser l'arithmétique de cette substitution sur le projet que ce livre étudie, car elle tient en 3 nombres. D'un côté, les capacités humaines, sur lesquelles la littérature de la relecture converge depuis 50 ans : une relecture qui trouve les défauts avance à 150-200 lignes par heure ; au-delà de 450 à 500 lignes par heure, la détection s'effondre ; une séance utile ne dépasse ni 400 lignes ni l'heure[8]. De l'autre, le flux : dans Darktable, le droit de fusionner s'est resserré jusqu'à un seul homme — les pull requests étaient fusionnées par une quinzaine de personnes au début des années 2010, cinq ou six en 2019-2020, deux ou trois depuis 2022, et un seul développeur en signe entre 95 et 100 % chaque année depuis 2019. L'année de pointe, 2020, il a fusionné 1 549 propositions. À la taille médiane mesurée — 156 lignes modifiées —, cela fait 242 000 lignes en un an, et c'est un plancher : à la taille moyenne mesurée (508 lignes), près de 800 000. Relire le seul plancher au rythme où l'on trouve les défauts demanderait plus de 3 heures et demie par jour, 365 jours par an — en sus d'un emploi à plein temps et de ses propres 600 commits annuels. Même au rythme d'effondrement, celui où l'on regarde encore mais l'on ne voit plus, il y faudrait une heure et quart chaque jour, à perpétuité. Les horodatages tranchent ce que

l'arithmétique laisse ouvert : un quart des fusions surviennent moins de 10 minutes après la précédente, et la fusion *moyenne* dépasse à elle seule en volume à relire le budget d'une séance de relecture maximale[9]. La conclusion décrit une architecture, pas un homme. Quand tout le code d'un projet passe par les soirées d'une seule personne salariée ailleurs, la relecture au sens de la littérature est arithmétiquement impossible, et il ne reste que le rituel : le tampon, le test vert, la confiance. L'échantillonnage le confirme au niveau des artefacts : sur 21 fusions récentes — toutes passées par les mêmes mains —, chacune porte bien une approbation formelle, et 19 ne portent *aucun* commentaire de relecture ligne à ligne. L'approbation existe ; elle est vide. Le goulot n'est pas un manque de vertu — c'est une conception d'organisation, et elle a les défauts d'une conception que personne n'a jamais faite.

Le culte du cargo n'est pas une pathologie moderne : c'est un régime très ancien, et la philosophie de la technique l'a cartographié. Jacques Ellul ouvrait son maître-livre en posant, après Mauss, que la magie est une technique au sens plein du mot — « on a eu raison », le redira-t-il 30 ans plus tard, « de considérer que la magie était aussi une technique »[10] — et il en décrivait l'appareil : des rites, des formules et des procédures fixés une fois pour toutes, parce que la moindre variation d'un mot ou d'un geste romprait l'équilibre magique — et pourquoi changerait-on un moyen qu'on tient pour le plus efficace ? La magie, écrit-il en endossant la formule de Masson-Oursel, est la première expression de la technique : une médiation vers des puissances qu'on ne contrôle pas, visant précisément à essayer de les

contrôler via des rituels codifiés, comme la technique matérielle est une médiation vers la matière. Et il ajoutait l'avertissement qui concerne directement ce chapitre : aujourd'hui encore, en étudiant les techniques qui s'adressent aux hommes, nous puisons sans le savoir dans le grand courant des techniques magiques[10]. Or les méthodes de développement, les processus, les bonnes pratiques récitées sont exactement cela — des techniques qui s'adressent aux hommes ; dès que leur raison se perd, elles retombent dans le régime d'origine : le rite fixe, exécuté pour sa forme, censé produire l'efficacité sur un domaine qu'on ne contrôle pas. Ellul décrivait même le mécanisme de défense du régime magique, et on jurerait qu'il décrit nos forums : quand le rite échoue, on blâme le magicien, jamais la technique. *Vous n'avez pas fait de l'agile correctement ; vous avez mal appliqué le pattern ; vous n'étiez pas assez SOLID.* Le rituel est infalsifiable par construction : ses échecs accusent l'officiant, jamais le rite. Voilà la loi générale que le chapitre « Poser le problème » annonçait, et elle a une suite : 20 ans plus tard, Ellul disait où mène ce régime quand il gagne — « ce n'est pas la technique qui nous asservit, mais le sacré transféré à la technique, qui nous empêche d'avoir une fonction critique »[10]. Une pratique sacralisée ne peut plus être critiquée, seulement récitée ou trahie ; le chapitre « Penser contre soi » retrouvera cette impossibilité de la critique à l'échelle d'un milieu entier.

Il existe une vieille règle anglaise pour les clôtures : n'enlevez jamais une barrière avant de savoir pourquoi on l'a plantée[11]. Le culte du cargo appelle la règle symétrique, qui est la loi de ce chapitre : n'appliquez jamais une règle sans savoir pourquoi elle a

été écrite. Les deux règles n'en font qu'une — dans les deux cas, l'unité de la pratique n'est pas la règle, c'est sa raison. C'est pourquoi, dans mon projet, les exigences imposées aux contributions — des fonctions qui ne font qu'une chose, une entrée et une sortie, deux niveaux d'imbrication au plus, des conditions complexes remisées dans des fonctions booléennes au nom lisible, des commentaires qui disent *pourquoi* et jamais *quoi* — sont publiées avec leurs raisons, et c'est pourquoi la règle finale de ces exigences autorise à suspendre chacune des autres : le jour où la règle contredit sa raison dans un cas donné, c'est la raison qui commande. Une discipline qui ne peut plus dire ses pourquoi n'est plus une discipline ; c'est une liturgie, et les liturgies ne font pas atterrir les avions. Cette loi vaut d'avance contre le présent livre : il se terminera sur une déontologie, et une déontologie est un objet dangereux — c'est la forme exacte que prend une règle quand elle s'apprête à devenir un rite. Chacun de ses articles sera donc donné avec sa raison, et aucun ne demandera qu'on le récite.

Et le chapitre suivant arrive au bon moment pour éprouver cette loi, car un instrument nouveau vient de bouleverser toutes les règles du métier, et il n'existe encore aucune liturgie établie pour lui : l'intelligence artificielle générative. Jamais le « pourquoi » n'aura été plus nécessaire que devant l'outil dont personne ne connaît encore les règles.

---

1. Feynman, R. P. (1974). « Cargo Cult Science ». *Engineering and Science*, 37(7), Caltech https://calteches.library.caltech.edu/51/2/CargoCult.htm —

discours de remise des diplômes de Caltech, juin 1974 ; repris dans *Surely You're Joking, Mr. Feynman !* New York : W. W. Norton, 1985 (trad. fr. *Vous voulez rire, monsieur Feynman !*, Paris : InterÉditions, 1985 ; rééd. Odile Jacob). La transposition au logiciel est postérieure et n'est pas de Feynman. ↩

2. Le terme « cargo-cult programming » est entré au Jargon File au début de 1991 (imprimé dans Raymond, E. S., dir., *The New Hacker's Dictionary*. Cambridge, MA : MIT Press, 1991) https://mitpress.mit.edu/9780262680691/the-new-hackers-dictionary/ : « un style de programmation dominé par l'inclusion rituelle de code ou de structures qui ne servent aucun but réel ». À l'échelle des organisations : McConnell, S. (2000). « Cargo Cult Software Engineering ». *IEEE Software*, 17(2), 11-13 https://stevemcconnell.com/wp-content/uploads/2017/08/CargoCultSoftwareEngineering.pdf . ↩

3. Hunt, A. & Thomas, D. (1999). *The Pragmatic Programmer : From Journeyman to Master*. Reading, MA : Addison-Wesley — le principe DRY y vise la duplication de *connaissance*, non de texte. Le repentir des auteurs, édition du 20e anniversaire (Addison-Wesley, 2020) https://pragprog.com/titles/tpp20/the-pragmatic-programmer-20th-anniversary-edition/ : « Dans la première édition, nous avons mal expliqué ce que nous entendions par *Don't Repeat Yourself*. Beaucoup ont compris que cela ne visait que le code : ne copiez-collez pas de lignes de source » (notre traduction). ↩ ↩

4. Metz, S. (2016). « The Wrong Abstraction ». sandimetz.com, 20 janvier 2016 https://sandimetz.com/blog/2016/1/20/the-wrong-abstraction — « prefer duplication over the wrong abstraction » ; « quand l'abstraction est fausse, le chemin le plus rapide en avant est en arrière ». La règle de trois : Fowler, M. (1999). *Refactoring : Improving the Design of Existing Code*. Reading, MA : Addison-Wesley https://martinfowler.com/books/refactoring.html, chap. 2, attribuée à Don Roberts (trad. fr. *Refactoring*, Paris : Dunod, 2019). ↩ ↩

5. Martin, R. C. (2008). *Clean Code : A Handbook of Agile Software Craftsmanship*. Upper Saddle River, NJ : Prentice Hall https://openlibrary.org/books/OL16900934M/Clean_code (trad. fr. *Coder proprement*, Pearson France). La critique : Muratori, C. (2023). « "Clean" Code, Horrible Performance ». computerenhance.com, 28 février 2023 https://www.computerenhance.com/p/clean-code-horrible-performance — critique de praticien, non revue par les pairs, sur l'exemple des formes du chapitre 6 de *Clean Code* ; facteurs mesurés d'environ 1,5× (sans polymorphisme) à 20-25× (tables et vectorisation). L'échange ultérieur entre Martin et Muratori est public (https://github.com/unclebob/cmuratori-discussion). ↩ ↩

6. Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (1994). *Design Patterns : Elements of Reusable Object-Oriented Software*. Reading, MA : Addison-Wesley https://www.informit.com/store/design-patterns-elements-

of-reusable-object-oriented-9780201633610 (trad. fr. *Design Patterns : catalogue de modèles de conception réutilisables*, Paris : Vuibert). La critique interne : Norvig, P. (1996). *Design Patterns in Dynamic Programming*, conférence Object World, 5 mai 1996 (norvig.com) https://norvig.com/design-patterns/ — « 16 des 23 patterns ont une implémentation qualitativement plus simple en Lisp ou Dylan qu'en C++ » ; Graham, P. (2002). « Revenge of the Nerds ». paulgraham.com https://paulgraham.com/icad.html — « When I see patterns in my programs, I consider it a sign of trouble. » ↩ ↩ ↩

7. Bacchelli, A. & Bird, C. (2013). « Expectations, Outcomes, and Challenges of Modern Code Review ». *Proceedings of ICSE 2013*, 712-721 https://doi.org/10.1109/icse.2013.6606617 — trouver des défauts est la motivation déclarée n° 1 ; les commentaires de défauts représentent un huitième de l'échantillon codé. Sadowski, C., Söderberg, E., Church, L., Sipko, M. & Bacchelli, A. (2018). « Modern Code Review : A Case Study at Google ». *Proceedings of ICSE-SEIP 2018*, 181-190 https://doi.org/10.1145/3183519.3183525 — « la revue a été introduite chez Google pour garantir la lisibilité et la maintenabilité du code » ; latence médiane inférieure à 4 heures. ↩ ↩

8. Fagan, M. E. (1976). « Design and Code Inspections to Reduce Errors in Program Development ». *IBM Systems Journal*, 15(3), 182-211 https://doi.org/10.1147/sj.153.0182 — inspection en équipe à 130-150 lignes par heure pour du code système (chiffres donnés comme prudents), séances

plafonnées à deux heures. Kemerer, C. F. & Paulk, M. C. (2009). « The Impact of Design and Code Reviews on Software Quality ». *IEEE Transactions on Software Engineering*, 35(4), 534-550 https://doi.org/10.1109/tse.2009.27 — le taux de 200 lignes par heure validé empiriquement : au-dessous, 56 à 66 % des défauts trouvés ; au-dessus, 45 à 51 %, en déclin continu à chaque palier de vitesse. Cohen, J. (2006). *Best Kept Secrets of Peer Code Review*. Austin, TX : Smart Bear https://search.worldcat.org/title/554483981 — l'étude Cisco (2 500 relectures) : détection optimale sous 300 lignes/heure, effondrement au-delà de 450-500, séance de 100 à 300 lignes et de 30 à 60 minutes ; étude publiée par un éditeur d'outils de relecture, non revue par les pairs, et citée ici avec cette étiquette. ↩

9. Propositions de code fusionnées du dépôt Darktable-org/Darktable, relevé exhaustif par l'API le 1er août 2026, champ `merged_by` — la personne qui clique la fusion, quel que soit le mode (commit de fusion, rebasage, écrasement) : 10 821 propositions fusionnées depuis 2012 ; 9 316 depuis 2019, dont 9 136 — 98 % — par le même développeur, avec un plancher annuel de 95 % (2020-2021) et 100 % à l'unité près depuis 2023. Année de pointe, 2020 : 1 549 fusions, une toutes les 5 heures et demie, nuits et dimanches compris. Tailles mesurées sur un échantillon de 100 diffs de premier parent des commits de fusion (50 récents, 50 de 2023) : médiane 156 lignes, moyenne 508, 90e centile 945, maximum 6 729. Les horodatages du sous-ensemble à commit de fusion (2 286 fusions, 2019-juillet 2026, réparties sur 1 275 jours)

montrent des pics du matin (7-9 h), du soir (17-19 h) et de la nuit (22-23 h), heure locale, et un quart des fusions à moins de 10 minutes de la précédente. Validation qualitative : sur 21 propositions fusionnées en 2026, les 21 l'ont été par le même développeur (aucune auto-fusion), latence médiane 11 heures ; 19 sur 21 sans aucun commentaire de relecture ligne à ligne. Ses propres commits : 599 sur les 12 derniers mois. ↩

10. Ellul, J. (1954). *La Technique ou l'Enjeu du siècle*. Paris : Armand Colin (rééd. Economica, 2008) https://archive.org/details/technique-ou-lenjeu-du-siecle-la-jacques-ellul, chap. I, section « Développement historique ». Les passages sur la magie sont restitués ici en discours indirect : leur texte a été vérifié sur la traduction anglaise (*The Technological Society*, New York : Knopf, 1964, p. 24-27 — « magic is a technique in the strictest sense of the word », après Marcel Mauss ; la fixité rituelle ; « the common reaction is to blame the magician rather than the technique »), non collationné sur l'édition française. La seule citation française directe est tirée d'Ellul, J., « Technique et Économie » (1984), in *Encyclopédie philosophique universelle*. Paris : PUF, 1989. La formule « première expression de la technique » est de Masson-Oursel, qu'Ellul endosse. Ellul distingue deux courants parallèles — technique matérielle et technique magique — qui se sont peu mêlés : l'application aux méthodes du logiciel est nôtre. La sacralisation (français vérifié) : Ellul, J. (1973). *Les Nouveaux Possédés*. Paris : Fayard (rééd. Mille et une nuits, 2003, p. 316) https://www.fayard.fr/livre/les-nouveaux-possedes-

9782842057824/ — « Ce n'est pas la technique qui nous asservit mais le sacré transféré à la technique, qui nous empêche d'avoir une fonction critique ». Le chapitre « L'instrument et la machine » reprendra Ellul sur l'outil et la machine. ↩ ↩ ↩

11. Chesterton, G. K. (1929). *The Thing*, chap. « The Drift from Domesticity » https://catholiclibrary.org/library/view?docId=/Contemporary-EN/XCT.165.html&chunk.id=00000011 — la « clôture de Chesterton » : ne retirez pas une clôture avant de savoir pourquoi on l'a plantée. ↩

14. L'intelligence en amont

Personne n'a avoué plus crûment où en est le métier de programmeur en 2026 que Kent Beck. Beck n'est pas n'importe qui : cosignataire du manifeste agile dont le chapitre « Poser le problème » a raconté la trajectoire, inventeur de l'*extreme programming*, artisan des pratiques qui ont gouverné le développement pendant 20 ans. Voici son bilan, en 2026 : « personne ne sait. Ce n'est pas seulement que moi je ne sais pas — personne ne sait »[1]. L'intelligence artificielle générative a balayé le recueil de pratiques de génie logiciel accumulé depuis deux décennies — le sien compris — et l'industrie est revenue, dit-il, en phase d'exploration : essayer tous azimuts, à petit coût, parce qu'aucune règle établie n'a survécu au changement d'outil[1].

Dans ce vide, deux discours occupent la scène, et il faut les lire de près, car aucun des deux ne dit ce qu'on lui fait dire. Le premier est la prophétie. Le patron d'Anthropic annonçait début 2025 que l'IA écrirait bientôt 90 % du code ; on cite toujours ce chiffre, jamais la phrase qui le suit immédiatement : « mais le programmeur doit toujours spécifier les conditions, l'application visée, les décisions de conception »[2]. Relisez la phrase omise : spécifier des conditions, viser une application, arrêter des décisions de conception — c'est, mot pour mot, ce que ce livre appelle concevoir. Même dans la bouche du prophète de l'automatisation, l'IA écrit du code ; elle ne décide pas de l'outil.

La conception reste humaine, et c'est tout le sujet de ce chapitre. Le second discours est une pratique, celle qu'Andrej Karpathy a baptisée *vibe coding* : « se laisser porter par les vibrations, embrasser les exponentielles, et oublier que le code existe » — accepter toutes les propositions de la machine sans lire les modifications, jusqu'à ce que, ce sont encore ses mots, le code grandisse au-delà de sa propre compréhension[3]. Karpathy réservait cela aux bricolages jetables du week-end ; la pratique s'est répandue bien au-delà, et le vocabulaire de l'année 2025 en enregistre le résultat : le dictionnaire Collins a couronné « vibe coding » pendant que Merriam-Webster couronnait « slop », la bouillie générée — la méthode et son produit, sacrés la même année[3]. Beck a décrit où mène cette pente, et il lui a donné un nom : l'iceberg. Ce que l'IA régénère volontiers, c'est la partie émergée d'un système — l'interface, la logique visible. Sous la ligne de flottaison vit tout ce que l'ancien code faisait sans le dire : les règles de conformité, les cas limites, 20 ans de leçons durcies en correctifs. Jeter l'iceberg en n'ayant regardé que sa pointe, c'est courir au-devant d'ennuis graves[1] — et l'on reconnaît le chapitre « Le code est un passif » : le coût du code n'a jamais été de l'écrire, mais de le comprendre.

Ce chapitre n'est donc pas une prophétie de plus — le terrain en regorge. C'est un chapitre de méthode : où cet instrument rend-il un concepteur meilleur, et sous quel protocole ?

D'abord, les faits, car ils sont contradictoires et la contradiction est instructive. Dans l'expérience contrôlée de GitHub, des développeurs recrutés pour implémenter un serveur web standardisé, à partir de zéro, allaient 55,8 % plus vite avec

l'assistant IA[4]. Dans l'essai contrôlé randomisé de METR, 16 développeurs chevronnés du logiciel libre, travaillant sur *leurs propres* dépôts — des projets mûrs, plus d'un million de lignes, 5 ans d'ancienneté moyenne — ont été *ralentis de 19 %* par l'IA sur 246 tâches réelles… tout en estimant, après coup, qu'elle les avait accélérés de 20 %[5]. Il faut lire les deux chiffres ensemble. Sur du terrain vierge et une tâche bien spécifiée, l'outil paie ; sur du code mûr qu'on connaît mieux que lui, l'outil coûte — et, dans les deux cas, *la perception ne mesure rien* : les développeurs de METR se croyaient plus rapides pendant qu'ils ralentissaient, exactement la préférence déclarée contre la préférence révélée du chapitre « Décider, puis tenir », appliquée cette fois aux ingénieurs. La conclusion n'est pas que l'outil est bon ou mauvais ; c'est qu'il n'a pas de valeur propre — sa valeur est celle du protocole qui l'encadre, et le reste de ce chapitre décrit le protocole.

Ma thèse tient en une phrase : l'intelligence artificielle sert à créer de la connaissance et de la compréhension de la tâche — et elle doit en rester là, *en amont*. Trois usages, dans l'ordre. Le premier est l'état de l'art. La recherche des méthodes existantes est l'étape préalable de tout travail de recherche et développement, et l'outillage classique s'est effondré : les moteurs de recherche généralistes réinterprètent les requêtes au point d'être devenus inutilisables pour le travail technique, et les moteurs académiques exigent de connaître d'avance les mots-clés exacts — le problème du vocabulaire du chapitre « Travailler pour le travail », appliqué au chercheur lui-même. Avant, la stratégie communément employée était une paresse rationnelle : prendre la méthode déjà implémentée dans sa bibliothèque logicielle,

faute de 3 ans devant soi pour prototyper les autres. Le modèle de langage (LLM) change la nature de l'étape : il pointe des méthodes candidates à partir d'une simple description du problème, puis — et c'est le renversement — *il les implémente et les compare toutes* sur votre banc d'essai. L'état de l'art cesse d'être une bibliographie, il devient une campagne de mesures.

Le deuxième usage en découle : balayer l'espace des solutions. Les méthodes numériques itèrent sur une solution jusqu'à convergence du résultat ; l'exploration assistée ouvre un axe d'itération au-dessus : itérer sur *les méthodes elles-mêmes*. Donnez des métriques d'erreur objectives contre une vérité terrain — pas d'impressions : des nombres, chapitre « Mesurer sans se mentir » — et l'assistant essaie les représentations du signal, les estimateurs, les variantes d'implémentation, formule des hypothèses sur ses propres échecs, les teste, se corrige, et revient avec le tableau complet[6]. Il faut mesurer ce que ce balayage est, et ce qu'il n'est pas : c'est l'inverse exact de l'entraînement d'un réseau de neurones. Le réseau apprend des poids ininterprétables — une boîte noire, biaisée comme son échantillon ; le balayage assisté, lui, « n'entraîne » rien : il *sélectionne*, mesures à l'appui, parmi des méthodes explicables — et ce qui reste à la fin est un modèle mathématique qu'un humain comprend, écrit noir sur blanc, réimplémentable dans 10 ans sans aucune IA. L'un accumule du passif opaque ; l'autre thésaurise de la connaissance.

Le troisième usage est le prototype, et il faut dire ce qu'il signifie pour un développeur de logiciel de bureau écrit en C — le pire langage qui soit pour ce travail, verbeux, dangereux, sans filet. En C, prototyper coûte si cher que le balayage de solutions se

réduisait au minimum vital : on implémentait *une* idée, la première défendable, et on vivait avec. Un laboratoire pouvait se payer mieux ; pas un homme seul. L'assistant rend au développeur solitaire l'économie d'un laboratoire : la maquette Python en une heure, le banc d'essai dans l'après-midi, 15 variantes mesurées et enterrées dans la semaine[6]. J'en ai fait la mesure sur un chantier daté : la réécriture du système d'historiques du chapitre « Le code est un passif », tri topologique restauré compris, m'a pris 3 jours au lieu des 3 semaines que j'estimais. Et il avale les corvées qui n'ont jamais mérité un cerveau : le code d'infrastructure répétitif, et l'archéologie — reconstruire l'arbre d'appels d'une fonction à travers 300 000 lignes, travail qui n'existe, le chapitre « Le code est un passif » l'a dit, que parce que le code n'est pas clos sur lui-même. J'en ai tiré un usage que je n'attendais pas : lui faire lire l'assembleur que le compilateur GCC tire de mon C. Vérifier ce que le compilateur a réellement fait — vectorisé ou non, fusionné ou non — est plus fiable que le chronométrage aveugle, qui mesure le bruit de la machine autant que le code ; on retrouve le chapitre « Mesurer sans se mentir » : lire l'artefact plutôt que deviner sur un chiffre agrégé. Tout ceci se résume d'un trait : employer du *high-tech* au moment de la recherche pour produire du meilleur *low-tech* au moment de l'exécution.

Ces chiffres suggèrent pourtant un homme 10 fois plus productif, et ce n'est pas ce qui s'est passé. La fatigue a changé d'objet ; elle n'a pas changé de volume. Celle de tout prototyper soi-même et de devoir se casser la tête sur tous les détails sémantiques et idiomatiques du langage utilisé a été remplacée par celle de relire

le volume de code qu'une machine produit en si peu de temps — et si l'on s'en tient à la seule règle défendable, ne verser au dépôt que du code qu'on a lu et compris, cette relecture coûte l'essentiel de ce que l'écriture aurait coûté. C'est le chapitre « Le code est un passif » retrouvé par l'autre bout : le coût du code est à la lecture et à la compréhension, et ce coût-là, l'assistant ne le supprime pas — il le concentre sur vous. On tient d'ailleurs là une part plausible du ralentissement mesuré par METR : les minutes gagnées à ne pas écrire se reperdent à relire. Le vrai gain n'est donc pas de quantité, il est de nature : l'attention quitte les détails d'implémentation — qu'il faudra de toute façon relire — pour se fixer là où rien n'est délégable : l'architecture, la méthode, les critères. On ne sort pas de la journée moins fatigué : la fatigue de faire s'est seulement transformée en fatigue de relire.

Reste le protocole, car la vitesse sans lui est exactement le *vibe coding*, et j'ai publié le mien avec les mathématiques qu'il a produites : la reconstruction des hautes lumières d'Ansel, réécrite en 2026 au terme d'une campagne homme-machine documentée de bout en bout[6]. Les règles tiennent en peu de lignes. Toute variante est implémentée dans la maquette en Python *et* dans le C, comparée d'une implémentation à l'autre, mesurée sur chaque scène synthétique à vérité terrain et sur des images naturelles, tabulée — et rien n'est retenu sans mon approbation explicite. Une seule autonomie est déléguée : si une variante dégrade *toutes* les métriques, l'assistant peut l'enterrer seul sans relecture humaine ; il ne peut jamais accepter seul. Ces deux règles ont suffi à balayer l'espace des solutions à vitesse machine sans que la conception m'échappe — et la division du travail qui en sort

mérite d'être dite, parce qu'elle est générale : à l'humain les intuitions physiques, les critères, et l'œil — des variantes ont gagné sur toutes les métriques et furent rejetées sur images, parce que les nombres ne voient pas tout ; à la machine les barres d'erreur, l'endurance, et cette froideur utile qui déclare votre hypothèse favorite mesurablement fausse du même ton égal qu'elle enterre les siennes. Les vices de l'instrument sont réels : il a ses méthodes de prédilection et déconseille les autres — j'ai dû forcer l'exploration de pistes qu'il désapprouvait, et l'une d'elles a gagné ; quand les métriques se contredisent, il tranche « au mieux » selon des critères qui ne sont pas les vôtres — d'où la règle du tableau complet, toujours ; il oublie des consignes, et il écarte en silence ce qu'il juge trop coûteux. Ce n'est pas un collègue — et ce n'est pas non plus un simple outil, car un outil ne travaille pas pendant que vous cuisinez, un outil ne prend pas d'initiative, un outil n'a pas d'opinion sur la forme que devrait prendre votre code. C'est un stagiaire : il ne connaît que ce qui s'apprend à l'école, propose des idées neuves et des absurdités avec le même aplomb, exige des consignes non ambiguës et une supervision constante — et ses erreurs sont sournoises, parce qu'enfouies au milieu d'un code par ailleurs parfaitement valide. Un stagiaire prodige et opiniâtre, donc ; ce chapitre lui donne un poste de travail et un règlement. Et le règlement répond à l'iceberg de Beck : si la partie immergée reste comprise, c'est qu'ici *rien ne passe sans être compris* — le protocole interdit précisément de jeter ce qu'on n'a pas expliqué. Quant au risque d'en désapprendre le métier — des systèmes que personne ne comprendra plus dans 10 ans, pendant qu'un tiers des employeurs disent déjà remplacer une partie de leurs débutants

par l'IA[1] —, la réponse est la même : le livrable de la campagne n'est pas le code, c'est l'article qui en fixe les mathématiques, jamais écrites nulle part auparavant, la documentation du code, les commentaires, l'organigramme. La connaissance est thésaurisée ; le code n'est que sa conséquence. Et la leçon comptable du chapitre « Le code est un passif » sort durcie de l'expérience : l'IA abaisse vers zéro le coût d'*écrire* du code, alors que le coût du code n'a jamais été à l'écriture, mais à la lecture, à la compréhension et à la maintenance — un stagiaire infatigable qui produit du passif à grande vitesse rend la parcimonie plus vitale, pas moins.

Reste le prix, à payer comptant plutôt qu'en promesses. Entraîner et faire tourner ces modèles coûte cher en énergie, et la littérature donne des ordres de grandeur plus nuancés que les slogans : le chiffre célèbre des « cinq voitures » ne concernait pas l'entraînement d'un modèle mais une campagne entière de recherche d'architecture, estimée et non mesurée, et l'estimation fut ensuite revue à la baisse d'un facteur considérable ; à l'autre bout, le seul grand modèle au bilan carbone intégralement publié — entraîné en France, sur un réseau nucléaire — pèse une cinquantaine de tonnes de CO_2 en cycle de vie complet, et c'est le réseau électrique, plus que l'algorithme, qui fait l'écart ; et l'inférence générative, elle, coûte des ordres de grandeur de plus que les traitements classiques, multipliée par des milliards de requêtes[7]. Est-ce que le low-tech produit en aval rembourse cette dépense d'amont ? Probablement pas, et je refuse de prétendre le contraire — d'autant qu'un vieux paradoxe attend au tournant : Jevons a montré en 1865 que l'usage plus économe

d'une ressource en *augmente* la consommation totale, « la vérité est tout le contraire » de l'intuition[8] ; l'efficacité rendue moins chère se dépense davantage : la forme extrême en est l'effet de rebond, qui enterre le fait que la frugalité ne peut être qu'une discipline (donc un choix politique) car elle ne sera jamais une loi du marché. Ce que je défends est plus étroit et tient debout : *à dépense faite*, il y a deux directions. Embarquer le high-tech dans le produit — l'inférence dans chaque export, la boîte noire dans chaque foyer — condamne à payer le coût à chaque exécution, chez chaque utilisateur, pour toujours. Le dépenser en amont pour concevoir des outils frugaux, explicables, durables et réparables — le critère low-tech tel que Bihouix l'a posé pour toute technique soutenable[9] — concentre le coût là où il n'est payé qu'une fois, et livre chez l'utilisateur un logiciel qui consomme *moins* qu'avant. C'est la seule forme de la dépense que je sache défendre : celle qui s'éteint.

D'où ma règle de conduite, et elle enveloppe tout le chapitre : j'utilise l'intelligence artificielle en supposant que je n'y aurai plus accès demain. Prix, licence, censure, faillite, régulation : l'accès est un robinet que d'autres tiennent, et le chapitre « Le code est un passif » a dit ce que valent les dépendances qu'on ne maîtrise pas. Tout ce que l'assistant produit doit donc survivre à l'assistant : des mathématiques écrites, des bancs d'essai qui tournent seuls, des protocoles documentés, du code standard sans couche tierce superflue — des flux de travail et des connaissances qui durent, utilisables sans IA, par moi ou par un autre, dans 10 ans. L'intelligence artificielle est un échafaudage : ce qu'on bâtit avec doit tenir sans elle. Le reste — l'outil fini, ce qu'il doit à celui

qui s'en sert — ne change pas d'un iota, et la réponse tient toujours en un mot qu'il faudra le chapitre suivant pour mériter : transparente.

---

1. Beck, K., entretien avec G. Orosz, « How Kent Beck Shapes the Software Engineering Industry ». *The Pragmatic Engineer Podcast*, 1er juillet 2026 https://newsletter.pragmaticengineer.com/p/how-kent-beck-shapes-the-software — « nobody knows. It's not just that I don't know. It's nobody knows » (transcription officielle, 2 h 14) ; « ce recueil de recettes a été effacé » ; l'« iceberg » du *vibe coding*, avec l'exemple d'un calcul de paie dont la conformité réglementaire est la partie immergée. Les phases d'exploration : Beck, K. (2016). « The Product Development Triathlon » https://medium.com/@kentbeck_7670/the-product-development-triathlon-6464e2763c46 et « Fast/Slow in 3X : Explore/Expand/Extract » https://medium.com/@kentbeck_7670/fast-slow-in-3x-explore-expand-extract-6d4c94a7539 . Medium. Le chiffre des débutants : GMAC (2026). *Corporate Recruiters Survey 2026* https://www.gmac.com/-/media/files/gmac/research/employment-outlook/2026-corporate-recruiters-survey/report.pdf — « un tiers des employeurs mondiaux déclarent avoir remplacé au moins une partie de leurs postes de débutants par l'intelligence artificielle » (621 recruteurs, 39 pays ; l'échantillon vise le recrutement des diplômés d'écoles de commerce). ↩ ↩ ↩ ↩

2. Amodei, D., « CEO Speaker Series ». Council on Foreign Relations, 10 mars 2025, transcription officielle https://www.cfr.org/event/ceo-speaker-series-dario-amodei-anthropic : « je pense que nous y serons dans trois à six mois — un monde où l'IA écrit 90 % du code […] Mais le programmeur doit toujours spécifier les conditions, […] l'application que vous essayez de faire, les décisions de conception d'ensemble » (notre traduction). ↩

3. Karpathy, A., message sur X, 2 février 2025 https://x.com/karpathy/status/1886192184808149383 : « There's a new kind of coding I call "vibe coding", where you fully give in to the vibes, embrace exponentials, and forget that the code even exists » ; plus loin : « I "Accept All" always, I don't read the diffs anymore. […] The code grows beyond my usual comprehension » — proposé, précise-t-il, pour les projets jetables du week-end. « Vibe coding », mot de l'année 2025 du Collins English Dictionary (6 novembre 2025) https://blog.collinsdictionary.com/language-lovers/collins-word-of-the-year-2025-ai-meets-authenticity-as-society-shifts/ ; « slop », mot de l'année 2025 de Merriam-Webster. https://www.merriam-webster.com/wordplay/word-of-the-year ↩ ↩

4. Peng, S., Kalliamvakou, E., Cihon, P. & Demirer, M. (2023). « The Impact of AI on Developer Productivity : Evidence from GitHub Copilot ». arXiv:2302.06590 https://arxiv.org/abs/2302.06590 — 95 développeurs recrutés, tâche standardisée (un serveur HTTP en JavaScript, à partir de zéro), 55,8 % plus rapide avec l'assistant ; prépublication

non revue par les pairs, et forte attrition (environ un tiers des recrutés au dénominateur final). ↩

5. Becker, J., Rush, N., Barnes, E. & Rein, D. (2025). « Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity ». METR, arXiv:2507.09089 https://arxiv.org/abs/2507.09089 — essai contrôlé randomisé, 16 développeurs, 246 tâches réelles sur leurs propres dépôts (22 000 étoiles et un million de lignes en moyenne, 5 ans d'ancienneté) : l'IA autorisée *allonge* le temps de 19 %, quand les développeurs prévoyaient 24 % de gain et en estimaient 20 % après coup. Instantané du début 2025 (outils : Cursor avec Claude 3.5 et 3.7), que METR présente lui-même comme daté. ↩

6. Pierre, A. (2026). « The Mathematics of Guided Laplacians and Harmonic Transposition ». ansel.photos, ressources https://ansel.photos/en/resources/guided-laplacian-highlights/ — la section « How this work was actually done » documente le protocole homme-machine complet (vérité terrain, doubles implémentations comparées, approbation humaine obligatoire, règle d'auto-rejet borné, documentation comme détecteur de bugs) et l'attribution des idées entre l'auteur et l'assistant ; la section « Where LLM shines » détaille l'état de l'art automatisé, le balayage de méthodes contre métriques objectives, et les biais de l'assistant. ↩ ↩ ↩

7. L'estimation des « cinq voitures » : Strubell, E., Ganesh, A. & McCallum, A. (2019). « Energy and Policy Considerations for Deep Learning in NLP ». *ACL 2019*, 3645-3650

https://doi.org/10.18653/v1/P19-1355 — le chiffre (626 155 livres de CO_2) porte sur une campagne complète de recherche d'architecture neuronale, non sur l'entraînement d'un modèle (l'entraînement d'un Transformer y pèse 192 livres) ; l'estimation fut recalculée à la baisse d'un facteur de plusieurs dizaines par Patterson, D. *et al.* (2021), arXiv:2104.10350 https://arxiv.org/abs/2104.10350. Le bilan mesuré : Luccioni, A. S., Viguier, S. & Ligozat, A.-L. (2023). « Estimating the Carbon Footprint of BLOOM ». *JMLR*, 24 https://www.jmlr.org/papers/v24/23-0069.html — 50,5 tonnes CO_2eq en cycle de vie complet, sur le réseau français à 57 g/kWh : c'est le réseau, plus que l'algorithme, qui fait l'écart avec les estimations d'autres modèles. L'inférence : Luccioni, A. S., Jernite, Y. & Strubell, E. (2024). « Power Hungry Processing : Watts Driving the Cost of AI Deployment ? ». *ACM FAccT 2024* https://doi.org/10.1145/3630106.3658542 — les tâches génératives coûtent des ordres de grandeur de plus que les traitements discriminatifs. ↩

8. Jevons, W. S. (1865). *The Coal Question*. Londres : Macmillan https://www.econlib.org/library/YPDBooks/Jevons/jvnCQ.html, chap. VII, p. 103 : « C'est une pure confusion d'idées que de supposer qu'un usage économe du combustible équivaut à une consommation diminuée. La vérité est tout le contraire » (notre traduction ; extraits français : *Futuribles*, n° 305, février 2005). Traitement moderne : Alcott, B. (2005). « Jevons' Paradox ». *Ecological Economics*, 54(1), 9-21. https://doi.org/10.1016/j.ecolecon.2005.03.020 ↩

9. Bihouix, P. (2014). *L'Âge des low tech : vers une civilisation techniquement soutenable*. Paris : Seuil, coll. « Anthropocène » https://www.seuil.com/ouvrage/l-age-des-low-tech-philippe-bihouix/9782021160727 (éd. revue, Points, 2021 ; trad. angl. *The Age of Low Tech*, Bristol University Press, 2020) — face aux limites de ressources, des techniques sobres, durables, réparables et à complexité maîtrisée, plutôt que la surenchère. ↩

15. L'outil transparent

Que doit la machine à celui qui s'en sert pour créer ? La réponse de ce chapitre tient en une image qu'il faut prendre au sérieux : le logiciel de traitement d'image est un laboratoire argentique virtuel. La chirurgie se pratique aujourd'hui à distance, un chirurgien manœuvrant un bras robotisé depuis une console ; l'interface d'un logiciel photo est exactement cela, le pupitre de commande d'un bras qui manipule un film simulé. Le film n'est pas dans un autre pays mais dans une autre dimension : celle de l'information pure, reconstruite à partir des données du capteur. L'utilisateur est dans le monde physique. Le logiciel est le portail qui relie ces deux mondes parallèles, et la commande à distance de processus réels simulés par et dans des flux d'information. De cette métaphore découle tout le reste : le rôle du logiciel n'est pas d'être intelligent ni de prendre des décisions à la place de l'artiste, comme un technicien de laboratoire virtuel ; il est de créer des axes de mouvement et des degrés de liberté pour un technicien *humain* qui opère à distance. J'appelle transparent l'outil qui remplit ce contrat : celui qui transmet l'intention sans y ajouter son avis, comme le pinceau transmet le geste. On ne demande pas son opinion à un pinceau, et que le pinceau soit incapable d'en donner une n'est pas un vice mais une vertu.

La conséquence esthétique de ce contrat est le cœur du chapitre : **l'esthétique s'obtient par la finesse du contrôle, pas par le goût de l'outil.** Et en fait de goût de l'outil, il faut s'entendre : il

s'agit du goût de ses concepteurs imprimé dans l'outil. Ce qui importe à un artiste n'est pas la beauté du résultat « sorti de boîte » ; c'est la distance qui sépare la sortie de l'algorithme du résultat qu'il avait en tête, et sa capacité à réduire cette distance en poussant les commandes. On reste dans l'idée maîtresse des premiers chapitres : refermer, au moins partiellement, le fossé qui sépare un état actuel d'un état préféré. Servir l'esthétique, pour un outil, c'est donc découper le processus en étapes élémentaires et offrir, à chaque étape, les degrés de liberté nécessaires pour piloter les compromis d'un pipeline imparfait. L'utilisateur conduit ; notre travail est que le volant tourne assez vite et que le levier de vitesses ne soit pas trop dur. Chaque étape ne fait qu'une chose, avec le moins d'effets de bord possible, car des étapes non orthogonales condamnent l'utilisateur à l'édition circulaire, où corriger le contraste dérègle la couleur qu'il faudra recorriger. Et le bon goût sorti de la boîte n'est pas un objectif de conception : « ça rend bien » est la responsabilité du pipeline entier, sous la supervision de l'artiste, jamais celle d'un algorithme isolé.

Car la phrase qui excuse tout, « ça rend bien », ne tient pas. Bien pour qui ? Dans quelles conditions, sous quels réglages, sur quels échantillons ? Les préférences esthétiques ne tombent pas du ciel : elles dépendent de l'*habitus*, ce système de goûts incorporés que la sociologie a montré socialement construit, où l'éducation artistique et l'entraînement de l'œil pèsent lourd[1]. Le « bien » du chercheur qui valide son algorithme n'est pas celui de la photographe qui a formé son regard sur 20 ans de tirages, et les chercheurs ont déjà déclaré « esthétiquement plaisants » des

résultats inutilisables pour un artiste formé. Les échantillons qui illustrent les publications sont choisis ; une horloge arrêtée donne l'heure juste deux fois par jour. Le monde académique du traitement d'image vit d'ailleurs de ce manège : trouver les cas limites de l'algorithme vedette, publier un algorithme *ad hoc* qui les corrige en créant les siens propres, et se faire moquer par l'article suivant, qui fera pareil. Aucun test empirique ne peut être assez exhaustif pour garantir qu'une méthode « marche » quand « marcher » se définit en goût subjectif. La sortie de ce manège est la discipline du chapitre « Poser le problème » : la qualité d'un algorithme se juge *uniquement* contre son objectif déclaré, ses contraintes et ses critères de réussite, formulés en propriétés vérifiables — pas de halos, conservation de la chromaticité, fidélité de reconstruction. Le test décisif d'un outil expressif est celui du **résultat connu d'avance** : donnez à un praticien un rendu qu'il connaît et qu'il vise ; le cahier des charges décide si l'outil peut l'atteindre, et l'essai le vérifie. Un outil se juge sur sa capacité à livrer ce que l'artiste savait vouloir, pas sur ce que l'outil propose à sa place.

Ces critères ne sont pas des vœux ; ils se testent. Un exemple concret : quand deux outils de retouche agissent l'un après l'autre sur une image, aucun ne doit introduire de teinte parasite là où il n'en touche pas — un dégradé qui devrait rester neutre ne doit pas virer au vert ou au magenta sur ses bords. On peut en faire un test automatique : passer une image synthétique dans l'algorithme et contrôler sa sortie. La conception rejoint ici la recherche : mon module de flou et de netteté est né d'un modèle physique de diffusion — l'équation de la chaleur poussée à un

ordre élevé, d'abord pour imiter des fondus d'aquarelle —, puis de la remarque qu'en inversant le signe du terme de diffusion, le même modèle *défloutait* au lieu de flouter. Une seule structure mathématique, deux usages opposés, obtenus proprement parce que le modèle avait un sens physique au lieu d'être un réglage empirique bricolé. J'avais auparavant tenté des méthodes de déconvolution plus ambitieuses ; je les ai abandonnées, non par manque d'intérêt, mais parce que leurs temps de calcul étaient irréalistes et leurs résultats imprévisibles — un outil dont on ne peut pas garantir le comportement n'est pas un outil transparent, quelle que soit son élégance théorique.

Sur quoi asseoir alors les modèles, si le goût est disqualifié comme fondement ? Sur la physique, car elle est la seule vérité de terrain disponible. Les pixels ne sont pas des données anonymes : ils encodent une émission lumineuse, le capteur enregistre de la lumière, l'écran en émet ; c'est un pipeline où la lumière est à la fois la matière entrante et la matière sortante. L'imagerie numérique a connu un accès d'orgueil en inventant des transformations de couleur anonymes, sans ancrage dans la physique des photons ni dans la psychophysique, pilotées par des paramètres sans unité, déconnectés du legs argentique. Résultat : le traitement numérique s'est construit *à côté* de l'héritage analogique au lieu de se construire dessus, et le savoir acquis d'un côté ne se transfère plus à l'autre, dans les deux sens, alors que les appareils photo ont toujours les mêmes vitesse, ouverture et sensibilité, et enregistrent toujours la même lumière. Pendant ce temps, l'image de synthèse prenait le chemin inverse, modélisant la physique des photons jusqu'au domaine spectral (et

plus seulement RGB), et produit aujourd'hui des rendus plus crédibles que la photographie numérique quand elle est mal post-traitée. Les peintres, eux, ont mélangé leurs pigments sur des bases purement physiques pendant des millénaires, sans nommer la chimie qu'ils exploitaient, et ont entraîné leur œil à prédire le résultat. La leçon d'architecture est celle du modèle-vue-contrôleur : que le modèle d'interaction soit physique n'empêche nullement le résultat d'être une perception totalement subjective — cela garantit seulement que ce que l'artiste manipule se comporte comme le monde dont son œil a l'habitude. À quoi s'ajoutent les exigences du chapitre précédent, version traitement du signal : des transformations continues, monotones, inversibles, au comportement asymptotique physiquement plausible ; et une propriété qui mérite son nom, l'échec gracieux. On sait d'avance que des paramètres poussés à fond produiront un résultat laid ; juger la qualité visuelle dans ces conditions n'apprend rien. Ce qui s'évalue, c'est la tenue des contraintes de conception même sous des réglages déraisonnables : l'algorithme qui échoue de façon prévisible et cohérente peut être amélioré ; celui qui échoue au hasard est bon à jeter : on n'améliore pas l'aléatoire.

Ces critères n'ont toutefois de valeur que si on les *lit*, et c'est ici qu'un test peut se retourner en son contraire. Une suite de tests mesurait, sur chaque modification du moteur, l'écart de couleur induit par rapport à une référence, avec un seuil de tolérance. Un jour, un correctif fut fusionné en mon absence sur un module de couleur dont j'étais l'auteur ; le test signalait un écart non nul. Or la nature mathématique du changement imposait un écart *exactement nul* sur l'image de test : le nombre non nul, si petit

fût-il, prouvait qu'une interaction anormale s'était produite ailleurs, et il aurait fallu chercher à comprendre. Le changement fut accepté parce que l'erreur moyenne sur la sortie du filtre d'image était sous le seuil, et donc a priori indolore. C'est la loi de Goodhart du chapitre « Mesurer sans se mentir » prise sur le fait : le seuil, conçu comme une alarme, servit de quitus ; personne ne chercha *pourquoi* un écart attendu à zéro ne l'était pas, puisque l'écart était sous la valeur d'alarme. Un critère de validation n'absout pas — il alerte ; le confondre avec un permis de passer, c'est transformer l'instrument de la rigueur en dispense de rigueur[2].

L'objection contemporaine est une école entière, et elle a le vent, l'argent et les processeurs pour elle : l'automatisation intégrale. Que l'outil décide, puisqu'il « sait » ; la magie en deux clics pour des utilisateurs qui rêvent d'un logiciel complexe qui les dispense d'un savoir-faire complexe ; l'IA qui applique le style appris. La réponse commence par dégonfler le vocabulaire, et Aaron Hertzmann l'a fait mieux que personne : les algorithmes d'apprentissage machine les plus performants sont de l'*ajustement de courbe glorifié* — caler des courbes de très haute dimension sur des millions de points[3]. Il n'y a pas de goût là-dedans ; il n'y a que la moyenne d'un corpus d'entraînement, c'est-à-dire l'habitus majoritaire de ceux qui ont produit et étiqueté les données, appliqué à votre image. La correction automatique n'est pas neutre : elle est l'opinion de quelqu'un d'autre, servie de force et sans signature. Cela ne condamne pas l'automatisme : personne ne regrette la mise au point automatique (autofocus). Cela lui assigne sa place contractuelle :

l'automatisme est recevable comme *proposition* — un point de départ que l'artiste peut inspecter, doser et refuser —, jamais comme verdict. D'où les deux clauses non négociables de l'outil transparent : chaque automatisme est doublé d'un interrupteur qui le débraye, et les hypothèses internes des algorithmes sont réduites au minimum, documentées, et modifiables à l'exécution — une porte de sortie pour l'utilisateur, et accessoirement pour le concepteur, le jour où l'une de ses hypothèses se révèle fausse. L'outil qui décide à la place de l'artiste et ne se laisse pas débrayer n'est plus un instrument : c'est un distributeur de résultats, et le résultat distribué est moyen, dans tous les sens du terme, par construction. On tient ici, sous forme de clause technique, un partage qui vaut bien au-delà de l'image : décider pour l'autre est une confiscation, le rendre capable de décider en est le contraire exact — et rien ne ressemble davantage à la première qu'un automatisme conçu, de parfaite bonne foi, pour le bien de son destinataire.

Ce chapitre a défini le contrat du logiciel expressif : télécommande, pas intelligence ; finesse du contrôle, pas bon goût en sortie d'usine ; critères vérifiables, pas « ça rend bien » ; physique comme base, perception comme juge ; échec gracieux ; débrayage universel. Mais le logiciel n'est qu'un cas récent d'une affaire beaucoup plus ancienne : ce qu'un instrument doit à l'artiste, la lutherie le sait depuis des siècles, et l'informatisation forcée des pratiques artistiques l'a fait oublier en une génération. C'est l'objet du chapitre suivant.

1. Bourdieu, P. (1979). *La Distinction : critique sociale du jugement*. Paris : Minuit https://catalogue.bnf.fr/ark:/12148/cb34628792q — le goût comme disposition socialement acquise. Le concept d'habitus est élaboré dans Bourdieu, P. (1972). *Esquisse d'une théorie de la pratique*. Genève : Droz https://doi.org/10.3917/droz.bourd.1972.01 . ↩

2. L'écart de couleur y est mesuré en ΔE (CIELAB), la distance perceptuelle standard entre deux couleurs. L'épisode est public : dépôt Darktable, *pull request* n° 12233 https://github.com/Darktable-org/Darktable/pull/12233 , août 2022 — un écart maximal de l'ordre de 1,6 ΔE accepté sous le seuil, là où la transformation exigeait 0, sans que l'anomalie sous-jacente soit recherchée. ↩

3. Hertzmann, A. (2018). « Can Computers Create Art ? ». *Arts*, 7(2), 18 https://doi.org/10.3390/arts7020018 : « Today, the most-successful AI and machine learning algorithms are best thought of as glorified data-fitting procedures. » ↩

16. L'instrument et la machine

> *« Un objet devrait faire tout ce qu'il peut faire, puis disparaître. […] Si un objet est profondément aimé, utilisé par habitude ou par attachement, ou transmis en héritage, l'objet a réussi sous sa forme la plus haute. »* — Manifeste anonyme circulant en ligne, source inconnue.

J'ai appris le piano, et je l'ai enseigné. Le paradigme y est connu de tous les élèves : *tais-toi et travaille* — où « travailler » se dit au sens de celui qui travaille sa gamme ou son trait : développer par l'exercice et par la théorie l'habileté et la compréhension de ce qu'on fait. Rien à voir avec l'injonction productiviste à s'user à la tâche. S'entraîner pour acquérir et entretenir l'agilité, répéter les mêmes mesures jusqu'à ce que le son soit juste, travailler la qualité, la texture et l'expressivité du son par le poids posé sur les touches, la vitesse de l'attaque, le mouvement du poignet et de l'avant-bras. Se servir de son corps pour la motricité fine et l'entraîner au-delà de ses capacités du moment, jusqu'à ce que des gestes parfaitement artificiels deviennent des réflexes. Et, à la fin, apprendre à entendre dans sa tête le son que l'on cherche avant même de bouger les mains. Rien de tout cela ne sera jamais réductible à des curseurs, des variables ou des algorithmes. C'est pourquoi je n'ai jamais compris la passion de tant de photographes, souvent les plus technophiles, pour les flux de travail presse-bouton. Tout se passe comme s'ils avaient été punis

par la photographie numérique, condamnés à passer par un logiciel pour obtenir leurs images, et qu'il fallait donc réduire cette corvée à une procédure sans âme, où tout ce qui peut être automatisé doit l'être. On s'échange des préréglages, certains en vendent, et l'intelligence artificielle applique désormais le style d'un autre. Où est la joie d'industrialiser la création artistique ? Quel intérêt y a-t-il à avoir un passe-temps qui pèse comme une corvée ? L'art est un processus, pas une procédure. Quel esprit malade concevrait un robot qui fasse la poterie, la gravure ou l'arrangement floral à votre place ? À quel âge perd-on l'appétit de faire les choses soi-même, même mal ?

Mais la punition est néanmoins réelle : les photographes n'ont pas choisi l'informatique, elle leur est tombée dessus. En 20 ans, le numérique a remplacé une technologie de 160 ans, connue et maîtrisée, et dans l'opération l'ordinateur est un pur *overhead* : un coût de structure qui ne produit rien par lui-même. Cette conscription — l'ordinateur imposé pour un travail qui se faisait très bien sans lui — a un coût humain que le chapitre « La violence à trois étages » détaille. Le photographe argentique n'était pas nécessairement tireur : il confiait développement et tirage à un laboratoire. Le numérique a remis le développement des négatifs bruts sur ses épaules, sans formation, ni en gestion de la couleur ni en informatique générale — ce qui a poussé une génération vers des logiciels simplifiés jusqu'au jouet, qui ont ensuite *défini* l'attente courante de ce qu'un logiciel photo doit être. Le tireur travaillait les mains dans le produit fini, papier et négatif physiquement accessibles, ouverts à tous les détournements (maquillage, virages, solarisations) y compris

ceux que les fabricants n'avaient pas prévus ; le numérique a chassé l'artiste du médium — l'image vit dans l'ordinateur comme donnée pure, seuls les ingénieurs savent y accéder, et l'artiste est confiné aux fonctions que l'interface veut bien offrir. Et le langage lui-même a été écrit par des informaticiens plutôt que par des scientifiques de la couleur, et moins encore par des photographes : le chapitre précédent a dit ce qu'il en a coûté. La sociologie du travail a documenté la version professionnelle de cette conversion forcée : dès la fin des années 1990, l'embauche des photojournalistes se met à exiger des compétences logicielles, et le métier bascule vers l'écran[1]. On comprend alors d'où vient l'appétit de presse-bouton du début de ce chapitre, et il faut lui rendre cette justice : c'est une réaction. L'informatisation a d'abord fusionné de force deux métiers, sommant le photographe de devenir aussi son propre tireur, sans formation et souvent sans goût pour cela ; et cette charge non désirée a créé le marché de son contraire. Le préréglage vend la décharge, sous la forme que Braverman appelait la *séparation de la conception et de l'exécution*, principe du travail industriel taylorisé[2] : quelqu'un d'autre a conçu le rendu, vous l'appliquez. L'extension de ce concept au loisir est la mienne, et elle donne la mesure de l'étrangeté du résultat : pour fuir une corvée imposée, un passe-temps en vient à reproduire, à la maison et sur son temps libre, la division du travail de l'usine — on a échappé au tirage forcé en devenant l'ouvrier d'exécution du goût d'un autre.

Il faut prendre la mesure exacte de ce qui vient d'être énoncé, car c'est le cœur de tout ce livre, et la photographie n'en est que le cas particulier que je connais du dedans. Ce qui est arrivé aux

photographes est arrivé à tout le monde, dans tous les métiers, et jusque dans les guichets de l'administration : un monde entier s'est artificialisé sans demander l'avis de ceux qui l'habitent, et il fabrique chaque jour des gens utilisateurs forcés qui n'ont pas choisi d'y vivre et qui doivent y vivre quand même. Quelqu'un doit les aider à le faire. C'est la thèse que défendra la conclusion de ce livre.

L'usine n'est pas une image en l'air ; c'est le fond historique de l'affaire. Lewis Mumford a fixé la distinction : l'outil est flexible — un couteau lisse le bois, le taille, le fend, force une serrure, visse au besoin — et se prête à la *manipulation* ; la machine est spécialisée dans une opération unique et tend à l'*action automatique*[3]. Jacques Ellul a décrit ce que le règne de l'outil supposait : dans la technique traditionnelle, l'outil étant insuffisant par nature, c'est l'habileté de l'ouvrier qui compensait — l'accent portait sur l'humain qui se sert de l'outil, non sur l'outil ; et il a décrit la machine moderne, si spécialisée qu'au premier changement de besoin elle ne vaut plus rien[4]. J'étendrais cette grille au logiciel par une synthèse : ce que l'informatisation a importé dans l'atelier de l'artiste, c'est la logique de la machine — le mode automatique, le préréglage, le rendu moulé en une opération, remplacé quand la mode change, comme on remplace une machine-outil devenue bonne à rien. Ce qu'elle a évincé, c'est la logique de l'outil : des commandes générales, et l'accent sur l'habileté de la main.

Contre la machine, le modèle positif existe, et la musique le connaît depuis toujours : l'instrument. Un instrument est un outil porté à sa forme extrême : générique (un piano joue tout),

transparent au sens du chapitre précédent, et exigeant — sa difficulté appartient à la musique, pas à sa facture. C'est très exactement le programme que je donne à mon logiciel : être un instrument d'expression visuelle qui laisse l'artiste développer sa propre interprétation du matériau brut, comme un instrument de musique laisse le musicien interpréter la partition, en donnant une large étendue expressive ; en contrepartie, une formation théorique et pratique de base est attendue de l'utilisateur, et la qualité visuelle ne sera pas sacrifiée pour aplanir la courbe d'apprentissage. La recherche en interfaces musicales a formulé l'idéal de conception avec une précision définitive : un droit d'entrée faible, *sans plafond de virtuosité* — et elle constate que la plupart des interfaces musicales simples à utiliser prennent, après un bref usage, un caractère de *jouet*[5]. Le mot n'est pas une coïncidence : c'est celui du chapitre « L'éloge du simple ». L'interaction instrumentale est d'ailleurs un modèle établi de la recherche en interfaces : l'instrument comme transducteur bidirectionnel entre l'utilisateur et l'objet du domaine[6] — la définition formelle de la télécommande du chapitre « L'outil transparent ». Et la philosophie des techniques donne au programme sa caution la plus inattendue : Simondon tenait l'automatisme pour « un assez bas degré de perfection technique » ; le vrai perfectionnement d'une machine correspond au contraire à ce que son fonctionnement recèle une *marge d'indétermination*[7]. Cette marge est précisément l'endroit où vit l'artiste. La machine fermée sur son fonctionnement prédéterminé ne peut donner que des résultats sommaires ; la machine ouverte a besoin de l'humain « comme les musiciens ont besoin du chef d'orchestre » — l'analogie musicale est dans le

texte —, ou plutôt comme l'instrument a besoin d'un musicien. Reste la pratique elle-même, qu'on présente volontiers comme le prix à payer avant d'avoir le droit de créer, des années de gammes comme un péage à l'entrée de la musique. Sennett montre que c'est un contresens : à son plus haut degré, la technique cesse d'être une activité mécanique, et c'est seulement une fois qu'on fait bien une chose qu'on devient capable de la sentir pleinement et de la penser profondément en la faisant[8]. L'entraînement ne précède pas l'expression : il la rend possible. L'instrument réussi finit par disparaître dans l'usage, comme le marteau du menuisier, qui n'existe pour lui que lorsqu'il casse[9] ; l'outil qui re-juge l'intention de l'artiste est, en ces termes, en panne permanente. Reste le test du pianiste, qui manque encore au logiciel : quand on peine sur un piano de concert qui sort de révision, on sait que l'instrument n'est pas en cause ; devant un logiciel, il est plus difficile d'identifier si la faute incombe à l'instrument ou à son opérateur — sauf si l'outil publie son contrat, ses critères et ses limites, comme l'exige le chapitre « L'outil transparent ». Les bons retoucheurs tirent un résultat de n'importe quel logiciel ; la différence est le temps que cela leur coûte, et ce temps est la mesure de l'instrument.

L'affaire a un nom politique, forgé par Illich. L'outil convivial est celui que « chacun peut utiliser, sans difficulté, aussi souvent ou aussi rarement qu'il le désire, *à des fins qu'il détermine lui-même* » ; entre l'homme et le monde, il est « conducteur de sens, traducteur d'intentionnalité »[10] — je ne connais pas de plus belle définition de l'instrument transparent. Et Illich a nommé l'inverse : « il y a monopole radical lorsque l'outil programmé

évince le pouvoir-faire de l'individu. » Transposé au logiciel photo : l'outil programmé qui évince le pouvoir-faire, c'est le préréglage et le style généré, qui tiennent dans la lettre de la définition.

L'objection qui reste debout est la plus sympathique : la démocratisation. Les automatismes ouvrent la photographie à ceux qui n'ont ni le temps ni le goût de l'entraînement ; qui suis-je pour leur barrer la porte ? L'objection a un précédent instructif. En 1906, John Philip Sousa publiait « The Menace of Mechanical Music » : les machines qui chantent et jouent du piano « à la place du talent, de l'intelligence et de l'âme humaines » allaient faire refluer « la marée de l'amateurisme », et le pays cesserait de chanter[11]. L'histoire s'est moquée de lui sur l'écoute : le disque a créé des auditeurs par millions, et des musiciens avec. Mais sur le geste, sa prédiction a mieux vieilli : le piano mécanique n'a pas créé de pianistes, et personne n'a jamais appelé le piano mécanique une démocratisation du piano. La distinction tranche l'objection : écouter est une consommation, créer est une production, et l'automatisation intégrale démocratise la première en évinçant la seconde. La réponse complète tient dans le critère des interfaces musicales : abaisser le droit d'entrée est légitime, et même souhaitable — c'est le plafond qui est le crime. Le préréglage comme point de départ inspectable, dosable, débrayable, abaisse l'entrée sans plafonner : c'est un instrument qui aide. Le mode automatique sans mode manuel plafonne l'expression à la moyenne de son corpus d'entraînement : c'est le monopole radical au sens strict, l'éviction du pouvoir-faire. La

démocratisation réelle d'un art passe par l'entrée, jamais par le plafond.

Concevoir pour l'art, c'est donc construire des instruments et refuser des machines : des outils génériques plutôt que des rendus moulés, une marge d'indétermination plutôt qu'un fonctionnement prédéterminé, des fins choisies par l'usager plutôt qu'exécutées pour lui, une entrée basse et aucun plafond. Reste la question que ce critère laisse ouverte, et elle est la plus délicate du livre : entre le rendu moulé qui impose le goût du développeur et la forêt de curseurs qui noie tout le monde, comment un instrument *ouvre*-t-il des possibles ? C'est l'objet du chapitre suivant.

---

1. Russial, J. & Wanta, W. (1998). « Digital Imaging Skills and the Hiring and Training of Photojournalists ». *Journalism & Mass Communication Quarterly*, 75(3) https://doi.org/10.1177/107769909807500312 ; Russial, J. (2000). « How Digital Imaging Changes Work of Photojournalists ». *Newspaper Research Journal*, 21(2), 67-83 https://doi.org/10.1177/073953290002100206 ; Klein-Avraham, I. & Reich, Z. (2016). « Out of the frame : A longitudinal perspective on digitization and professional photojournalism ». *New Media & Society*, 18(3), 429-446 https://doi.org/10.1177/1461444814545289. Ces études établissent le fait de la conversion forcée et du déplacement du travail vers l'écran ; le jugement porté ici est le mien. ↩

2. Braverman, H. (1976). *Travail et capitalisme monopoliste : la dégradation du travail au XX^e siècle*. Paris : Maspero (éd. orig. 1974), chap. 4 — les principes tayloriens, dont la séparation de la conception et de l'exécution https://openlibrary.org/works/OL1986011W . Braverman traite du travail salarié industriel ; l'extension au loisir est de l'auteur. ↩

3. Mumford, L. (1934). *Technics and Civilization*. New York : Harcourt, Brace & Co., p. 9-12 https://openlibrary.org/works/OL1879251W ; trad. fr. *Technique et Civilisation*. Paris : Seuil, 1950 ; rééd. Marseille : Parenthèses, 2016. ↩

4. Ellul, J. (1954). *La Technique ou l'Enjeu du siècle*. Paris : Armand Colin ; rééd. Economica, 2008 https://openlibrary.org/works/OL1985863W — chap. II sur les caractères de la technique traditionnelle, restitué ici en discours indirect : le texte a été vérifié sur la traduction anglaise (*The Technological Society*, 1964, p. 64-77 ; la machine mono-tâche p. 74-75), non collationné sur l'édition française. La répartition outil/machine entre Mumford et Ellul est respectée à dessein : Ellul refusait explicitement l'assimilation de la technique à la machine, et reprochait précisément à Mumford de la commettre ; la synthèse des deux est la mienne. ↩

5. Wessel, D. & Wright, M. (2002). « Problems and Prospects for Intimate Musical Control of Computers ». *Computer Music Journal*, 26(3), 11-22

https://doi.org/10.1162/014892602320582945 — la section s'intitule exactement « Low "Entry Fee" with No Ceiling on Virtuosity » ; le caractère « de jouet » des interfaces musicales simples y est constaté verbatim. ↩

6. Beaudouin-Lafon, M. (2000). « Instrumental Interaction : An Interaction Model for Designing Post-WIMP User Interfaces ». *Proceedings of CHI 2000*. New York : ACM, 446-453 https://doi.org/10.1145/332040.332473 . ↩

7. Simondon, G. (1958). *Du mode d'existence des objets techniques*. Paris : Aubier, Introduction https://openlibrary.org/works/OL4504172W — « le véritable perfectionnement des machines […] correspond non pas à un accroissement de l'automatisme, mais au contraire au fait que le fonctionnement d'une machine recèle une certaine marge d'indétermination », l'automatisme étant « un assez bas degré de perfection technique » ; l'homme y est l'organisateur permanent d'une société de machines ouvertes « qui ont besoin de lui comme les musiciens ont besoin du chef d'orchestre ». ↩

8. Sennett, R. (2010). *Ce que sait la main : la culture de l'artisanat*. Paris : Albin Michel (éd. orig. *The Craftsman*, Yale University Press, 2008), 1re partie, chap. 1 https://openlibrary.org/works/OL1870880W . Sur la mesure des 10 000 heures, Sennett s'appuie sur les travaux d'Ericsson (Ericsson, K. A. *et al.*, 1993, *Psychological Review*, 100(3), 363-406) https://doi.org/10.1037/0033-295X.100.3.363 . ↩

9. Winograd, T. & Flores, F. (1989). *L'intelligence artificielle en question*. Paris : PUF (éd. orig. *Understanding Computers and Cognition*, Ablex, 1986), chap. 3 https://openlibrary.org/works/OL28171792W — la reprise du marteau de Heidegger : disponible-à-la-main, l'outil disparaît de la conscience ; il ne redevient présent qu'à la panne (*breakdown*). ↩

10. Illich, I. (1973). *La convivialité*. Paris : Seuil (version française établie par l'auteur avec L. Giard et V. Bardet ; éd. orig. *Tools for Conviviality*, Harper & Row, 1973), chap. II pour la définition de l'outil convivial, chap. III pour le monopole radical https://openlibrary.org/works/OL2848898W . ↩

11. Sousa, J. P. (1906). « The Menace of Mechanical Music ». *Appleton's Magazine*, 8, 278-284 https://ocw.mit.edu/courses/21m-380-music-and-technology-contemporary-history-and-aesthetics-fall-2009/18ab3aba9fe7aa1502a55cd049333659_MIT21M_380F09_read02_sousa.pdf . Son pronostic sur l'écoute s'est révélé faux, celui sur la pratique amateur du geste musical, beaucoup moins. ↩

17. Ouvrir sans tenir le pinceau

Le chapitre précédent a laissé le concepteur d'instruments sur une arête étroite, entre deux précipices. D'un côté, tenir le pinceau à la place de l'artiste : le rendu moulé, le mode automatique, le préréglage-destination — autant de façons d'embarquer dans l'outil l'opinion du développeur sur ce qu'une image doit être, et de la servir à des milliers de gens qui ne l'ont pas demandée. De l'autre, refuser de choisir quoi que ce soit : déverser les paramètres bruts du moteur, et noyer tout le monde sous la forêt de curseurs dont le chapitre « L'éloge du simple » a fait le procès. La plupart des logiciels d'image oscillent entre les deux, et beaucoup cumulent : des automatismes opaques *et* 40 réglages. Ce chapitre cherche la crête — comment un instrument *ouvre* des possibles sans imposer un goût ni exiger un ingénieur — et il commence par instruire le dossier : qu'a fait l'ordinateur à la création, au juste ?

Ce qu'il a retiré, l'artisanat l'avait nommé avant l'informatique. David Pye distinguait deux régimes de facture : l'ouvrage *de risque*, où la qualité du résultat n'est pas déterminée d'avance mais dépend du jugement, de la dextérité et du soin que le faiseur exerce *pendant* qu'il travaille — la qualité continûment en jeu tout au long du geste — et l'ouvrage *de certitude*, celui de la production en série, où le résultat est réglé avant que le geste commence, et dont l'automatisation intégrale est la forme pure[1]. Le bouton-poussoir est l'ouvrage de certitude appliqué à l'art : le

résultat préexiste au geste, et le geste ne peut plus rien lui apporter — ni rien lui coûter. Or ce que le risque emportait avec lui, c'était la présence entière du faiseur. Sherry Turkle a documenté le deuil chez les professionnels du dessin quand la simulation s'est imposée : les architectes du MIT voulaient préserver des « espaces sacrés » — le dessin à la main d'abord — là où se jouait leur intimité avec l'objet[2]. Et Brian Eno a dit la même chose en musicien, dans une formule restée célèbre : « le problème des ordinateurs, c'est qu'il n'y a pas assez d'Afrique dedans » — et il précisait, moins cité : l'ordinateur « utilise si peu de mon corps. Vous êtes assis là, et c'est assez ennuyeux. Cette stupide petite souris qui réclame une main, et vos yeux. C'est tout. Et le reste de vous ? […] C'est une prison »[3]. Trois témoins, 3 métiers, un seul constat : l'informatisation de la création a d'abord été une amputation — du risque, de la main, du corps.

Elle a ensuite été une normalisation, et ce dossier-là commence avant l'ordinateur, en 1888 très exactement. « Vous appuyez sur le bouton, nous faisons le reste » : le slogan est de George Eastman lui-même, pour le premier Kodak — l'appareil se renvoyait entier à l'usine, qui développait, tirait et rechargeait[4]. C'est la première promesse presse-bouton de l'histoire de l'art, et son bilan est double, exactement comme celui du chapitre « L'instrument et la machine » : elle a créé la photographie populaire — des millions d'images, des millions de photographes — et elle a créé du même geste le *vernaculaire* de l'instantané : les mêmes cadrages, les mêmes poses, les mêmes sujets, une culture Kodak dont l'ethnographie a montré la stupéfiante uniformité[4]. Vilém Flusser a donné la théorie de ce phénomène, et elle est devenue le

texte canonique de la photographie : l'appareil est un programme, et « certes l'appareil fait ce que le photographe veut, mais le photographe ne peut vouloir que ce que peut l'appareil » — toutes les images qu'il produira sont contenues d'avance dans le programme, « prévisibles, redondantes »[5]. Le photographe ordinaire est le fonctionnaire de ce programme ; le photographe expérimental est celui qui « joue contre l'appareil », qui s'efforce de lui arracher une image *qui ne figure pas dans son programme* — « être libre, c'est jouer contre les appareils »[5]. Lev Manovich a prolongé Flusser dans l'ère du logiciel : les opérations du logiciel de création deviennent les opérations de la culture elle-même, et la question de conception se formule dans ses termes — comment le logiciel rend certains choix « naturels et faciles à exécuter, tout en cachant d'autres possibles »[6]. Le filtre de la plateforme fait le style de la décennie. Le faisceau est cohérent : quand des millions de créateurs reçoivent le même programme, les mêmes défauts d'usine et les mêmes préréglages, l'espace immense des images possibles se replie sur le petit sous-espace que l'outil rend facile — un rétrécissement qu'on a mesuré ailleurs, dans la musique populaire[7]. Le goût du développeur, multiplié par la distribution, devient le style d'une époque — et personne ne l'a choisi.

Que fait le bouton-poussoir à la créativité elle-même ? La psychologie fournit deux mécanismes, et ils convergent. Margaret Boden décrivait la création comme l'exploration, puis la transformation, d'un espace conceptuel : la créativité *combinatoire* recombine des éléments connus ; l'*exploratoire* parcourt un espace de possibles existant ; la *transformationnelle* — la plus rare — modifie les règles qui définissent l'espace lui-

même[8]. Lisez la hiérarchie en termes d'outils : l'outil définit l'espace que l'artiste peut explorer. Le bouton-poussoir réduit la création à parcourir un catalogue pré-énuméré — la forme la plus pauvre du premier niveau ; l'instrument ouvert permet l'exploration ; et seul l'instrument dont on peut déborder le programme — Flusser encore — laisse une chance à la transformation. Le second mécanisme est celui de Csikszentmihalyi : le plaisir de l'activité « apparaît à la frontière entre l'ennui et l'anxiété, quand les défis s'équilibrent avec la capacité d'agir » ; le défi trop bas ne produit pas du confort mais de l'ennui, puis le désengagement, le défi trop haut produit de l'anxiété et le sentiment d'être dépassé, l'optimum entre les deux est nommé le *flow* : un état presque second où la perception du temps est altérée, la fatigue ne se sent pas, le sujet est absorbé entièrement dans sa tâche et en éprouve une joie particulière.[9] Le presse-bouton est en dessous du seuil : il ne demande rien, donc il n'engage rien — c'est le « plafond de virtuosité » des interfaces musicales du chapitre « L'instrument et la machine », retrouvé par la psychologie. Il faut pourtant garder l'autre moitié du tableau, car elle est contre-intuitive : la page entièrement blanche ne libère pas non plus. La recherche sur les contraintes le montre : devant une liberté totale, on refait ce qui a déjà marché ; c'est la contrainte *choisie* qui force à quitter les réponses fiables et pousse vers les réponses neuves — Monet, Stravinsky et Chanel ont créé *par* leurs contraintes, pas malgré elles[10]. Toute la question est de savoir qui choisit la contrainte : celle que l'artiste s'impose est un moteur ; celle que le vendeur d'outil impose est un moule. Eno, qui a passé sa vie des deux côtés, a écrit la synthèse : « l'ennui commence avec une philosophie de

conception qui assimile *plus d'options* à *plus de liberté* » — le rapport à l'instrument, disait-il aux fabricants de synthétiseurs, se gagne parfois en *réduisant* les options[10].

De ce dossier, l'éthos se déduit terme à terme, et c'est la doctrine de mon propre logiciel. Premier principe : des *axes* de possibilités, pas des rendus. Un instrument honnête n'offre pas un catalogue de résultats — il offre les dimensions qui engendrent les résultats. La différence n'est pas de quantité mais de géométrie : un catalogue est fini, parcourable, normalisant ; un espace à quelques axes est continu, et ses points n'appartiennent à personne. C'est ici que se résout le dilemme des options du début : le remède à la forêt de curseurs n'est pas moins de contrôle, c'est l'*orthogonalité* — des dimensions perceptuellement calibrées, qui ne se battent pas entre elles, chacune faisant une chose que l'œil sait nommer. Quarante curseurs qui interfèrent sont une punition ; 4 axes orthogonaux sont un instrument — et construire ces axes, calibrer leurs échelles sur la perception plutôt que sur la mécanique interne, est précisément le travail de conception que le presse-bouton escamote et que la forêt de curseurs n'a pas fait. Deuxième principe : le point de départ est *fidèle*, jamais esthétique. L'outil livre d'abord la scène telle que mesurée, mise en forme par la physique et la perception — le contrat du chapitre « L'outil transparent » — et toute intention d'image appartient à ce qui suit, qui n'appartient qu'à l'artiste. Troisième principe : le préréglage est un exemple, pas une destination — le chapitre « L'instrument et la machine » a posé ses 3 conditions, inspectable, dosable, débrayable ; j'ajoute la quatrième : *pédagogique* — un préréglage devrait s'ouvrir et se lire, comme

une partition, pour qu'on apprenne de lui le chemin plutôt que d'y camper. Quatrième principe, et c'est le garde-fou de tout le reste : l'instrument n'énonce aucune esthétique — il rend les esthétiques *atteignables*. Sa réussite ne se mesure pas aux images qu'il favorise, mais à celles qu'il n'interdit pas : la marge d'indétermination de Simondon, l'entrée basse sans plafond de Wessel et Wright, le programme qu'on peut déborder de Flusser — 3 noms pour la même exigence. Concevoir pour l'art, c'est écrire le programme le plus riche possible et s'effacer devant qui le joue — ouvrir le champ, sans tenir le pinceau. Et cette exigence n'a rien de propre à l'art : rendre l'autre capable de ce qu'il n'aurait pas pu seul, puis s'effacer, est ce qu'on doit à quiconque dépend d'un outil. L'artiste est seulement le cas où l'enjeu se voit le mieux.

L'instrument qui y parvient fait tout ce qu'il peut faire, puis disparaît dans l'usage ; et s'il est profondément aimé, utilisé par habitude, transmis, il a réussi sous sa forme la plus haute. Mais un instrument ne se transmet que s'il dure, et un logiciel ne dure que si quelqu'un le finit et l'entretient. C'est l'objet du chapitre suivant.

---

1. Pye, D. (1968). *The Nature and Art of Workmanship*. Cambridge : Cambridge University Press (éd. révisée, Londres : Herbert Press, 1995), chap. 2 https://www.worldcat.org/title/nature-and-art-of-workmanship/oclc/32411126 — « la qualité du résultat n'est pas prédéterminée, mais dépend du jugement, de la dextérité et du soin que le faiseur exerce pendant qu'il travaille […] la

qualité du résultat est continûment en jeu pendant la fabrication » (*workmanship of risk*) ; l'ouvrage de certitude, « qu'on trouve toujours dans la production en quantité, et à l'état pur dans l'automation intégrale » (notre traduction). Pas de traduction française. ↩

2. Turkle, S. (2009). *Simulation and Its Discontents*. Cambridge, MA : MIT Press, p. 4-5 https://mitpress.mit.edu/9780262546799/simulation-and-its-discontents/ — les enseignants du MIT des années 1980 voulaient préserver des « sacred spaces » : « les architectes voulaient préserver le dessin à la main » (notre traduction). Le terrain est celui de professionnels du MIT (architecture, ingénierie, sciences), non d'artistes au sens strict ; le mécanisme décrit — la perte d'intimité avec l'objet quand la simulation s'interpose — est ce qui se transpose. ↩

3. Kelly, K. (1995). « Gossip is Philosophy », entretien avec Brian Eno. *Wired*, 3.05, mai 1995 https://www.moredarkthanshark.org/eno_int_wired-may95.html : « The problem with computers is that there is not enough Africa in them » ; la version « the trouble with computers », qui circule partout, est une déformation. Le passage sur le corps : « It uses so little of my body. You're just sitting there […] That's it. What about the rest of you ? […] It's imprisoning » (notre traduction). ↩

4. Le slogan est d'Eastman lui-même, pour l'appareil « The Kodak » de 1888 (le « No. 1 » est son successeur de 1889) : l'appareil, vendu préchargé de 100 vues, retournait entier à Rochester pour développement, tirage et rechargement.

L'ethnographie du vernaculaire : Chalfen, R. (1987). *Snapshot Versions of Life*. Bowling Green, OH : Bowling Green State University Popular Press https://search.worldcat.org/title/snapshot-versions-of-life/oclc/608210070 — la « culture Kodak » : conventions de cadrage, de pose et de sujet d'une uniformité remarquable à travers les albums de famille. ↩ ↩

5. Flusser, V. (1996). *Pour une philosophie de la photographie* (trad. J. Mouchard, depuis l'édition allemande révisée de 1989). Belval : Circé (rééd. poche, 2004) https://www.babelio.com/livres/Flusser-Pour-une-philosophie-de-la-photographie/631558 , p. 20 : « Certes l'appareil fait ce que le photographe veut, mais le photographe ne peut vouloir que ce que peut l'appareil » — d'où des images « prévisibles, redondantes » ; p. 29 : le photographe « ne joue pas avec son appareil mais contre lui » ; p. 83 : « Être libre, c'est jouer contre les appareils » ; p. 84 : les photographes expérimentaux s'efforcent « de tirer de leur appareil et de mettre en image quelque chose qui ne figure pas dans son programme ». Verbatims relevés dans les études flussériennes (Flusser Studies) citant l'édition Circé ; à collationner sur l'exemplaire avant impression. ↩ ↩

6. Manovich, L. (2001). *The Language of New Media*. Cambridge, MA : MIT Press https://mitpress.mit.edu/9780262632553/the-language-of-new-media/ (trad. fr. *Le Langage des nouveaux médias*, trad. R. Crevier, Dijon : Les Presses du réel, 2010) — chap. « Operations » : les opérations du logiciel deviennent des opérations culturelles générales. Manovich, L. (2013).

Software Takes Command. New York : Bloomsbury (pas de traduction française) https://doi.org/10.5040/9781472544988 — le logiciel comme « moteur universel » de la culture ; la question de conception, verbatim de « Inside Photoshop » (*Computational Culture*, 1, 2011) http://computationalculture.net/inside-photoshop/ : « comment le logiciel de création façonne-t-il les médias produits, en rendant certains choix naturels et faciles à exécuter, tout en cachant d'autres possibles ? » (notre traduction). Sur l'esthétique des plateformes : Manovich, *Instagram and Contemporary Image* (en ligne, 2016-2017) https://manovich.net/index.php/projects/instagram-and-contemporary-image . ↩

7. Serrà, J., Corral, Á., Boguñá, M., Haro, M. & Arcos, J. L. (2012). « Measuring the Evolution of Contemporary Western Popular Music ». *Scientific Reports*, 2, 521 https://doi.org/10.1038/srep00521 — 464 411 enregistrements, 1955-2010 : homogénéisation mesurable de la palette timbrale, restriction des transitions de hauteurs, montée du volume. L'étude constate ces tendances sans en identifier les causes ; l'attribution d'une part aux outils de production est notre hypothèse, non la leur. ↩

8. Boden, M. A. (2004). *The Creative Mind : Myths and Mechanisms*, 2e éd. Londres : Routledge (1re éd. 1990) https://books.google.com/books/about/The_Creative_Mind.html?id=6Zkm4dz32Y4C — créativité combinatoire, exploratoire, transformationnelle ; l'espace conceptuel dont

les règles définissent ce qui peut être pensé. Pas de traduction française. ↩

9. Csikszentmihalyi, M. (1990). *Flow : The Psychology of Optimal Experience*. New York : Harper & Row https://openlibrary.org/books/OL18354762M/Flow , p. 52 : « le plaisir apparaît à la frontière entre l'ennui et l'anxiété, quand les défis s'équilibrent exactement avec la capacité d'agir de la personne » (notre traduction ; l'édition française, *Vivre*, Paris : Robert Laffont, 2004, est une adaptation — nous retraduisons de l'anglais). Le défi inférieur à la compétence produit l'ennui ; défi et compétence bas ensemble, l'apathie (chap. 7). ↩

10. Stokes, P. D. (2005). *Creativity from Constraints : The Psychology of Breakthrough*. New York : Springer Publishing Company https://books.google.com/books/about/Creativity_from_Constraints.html?id=XkkmQo10ZVgC — les contraintes appariées : l'une exclut les solutions fiables, l'autre oriente vers les solutions neuves ; études de cas sur Monet, Picasso, Stravinsky, Chanel. Eno, B. (1999). « The Revenge of the Intuitive ». *Wired*, 7.01, janvier 1999 https://www.moredarkthanshark.org/eno_int_wired-jan99.html : « The trouble begins with a design philosophy that equates "more options" with "greater freedom" » ; dès l'entretien de 1995, il disait aux fabricants de synthétiseurs que l'enjeu n'était pas d'augmenter le nombre d'options internes, et que le rapport à l'instrument « pourrait être

mieux servi en réduisant les options » (notre traduction). ↩ ↩

18. Finir et maintenir

Le chapitre « Poser le problème » a laissé 3 des 5 lettres de la méthode SMART en attente : *assignable*, *réaliste*, *borné dans le temps*. Elles se ramènent à une seule règle, qui ferme cette partie du livre : **ne rien entreprendre qu'on n'ait les moyens de finir et de maintenir.** Le chapitre « L'éloge du simple » a établi qu'un outil se développe jusqu'à ce qu'il remplisse son objet ; ce chapitre dit ce que remplir son objet exige. Car finir n'est pas un état qui advient quand on cesse de travailler : c'est un objet de conception à part entière, qui se spécifie, se budgète et se protège — et la plupart des logiciels qui échouent n'échouent pas pour avoir été mal commencés, mais pour n'avoir jamais été finis.

Car commencer n'a jamais été le problème : commencer est une euphorie. Le dépôt neuf n'a ni dette, ni bugs, ni utilisateurs, ni histoire ; le plan paraît parfait parce que rien ne l'a encore éprouvé, et le code coule — Brooks ouvrait son livre là-dessus, les joies du métier, le plaisir de bâtir des châteaux de pure pensée[1]. Cette euphorie est un piège à deux mâchoires. La première : elle est auto-récompensée, quand finir ne l'est pas — le chapitre « Ce que l'entre-soi produit » montrera une salle qui applaudit les annonces et ignore l'entretien ; on peut donc passer une vie à commencer des projets, porté d'euphorie en euphorie, sans jamais rien livrer de fini, et s'y sentir productif. La seconde : ce qui est difficile n'est pas de lancer, c'est de *gérer la croissance* — et cela ne se voit qu'après. Les méthodes de travail qui suffisaient à

un homme seul sur 10 000 lignes s'effondrent à 100 000 avec des contributeurs ; l'infrastructure improvisée devient le goulot d'étranglement ; la communication informelle sature, chapitre « Le comité n'inventera rien ». Faire grandir un projet, c'est reconcevoir régulièrement le projet lui-même — ses outils, ses méthodes, son organisation — pendant qu'il roule, et cette conception-là n'a ni la gloire du commencement ni le confort de la page blanche. La mesure d'un ingénieur n'est pas ce qu'il démarre : c'est ce qu'il fait tenir debout dans la durée.

Première exigence : on ne programme pas une solution, on développe une architecture. Dans mon logiciel, derrière le module de débruitage — retirer le grain numérique d'une photo prise dans l'obscurité —, il y a un réseau de neurones qu'il faut entraîner sur des milliers d'images. Le réseau n'est que la partie visible. Autour, j'ai bâti une chaîne d'administration complète : un tri des images côté utilisateur, pour ne pas récolter n'importe quoi dans le corpus d'entraînement et filtrer en amont ; une validation des contributions côté mainteneur ; un réentraînement automatisé du modèle qui peut être accompli par n'importe que en cliquant simplement sur un bouton. L'architecture du réseau lui-même n'a rien de novateur, et le code intégré au logiciel est modeste ; ce qui fait tenir l'ensemble, c'est la chaîne de travail autour, celle qui évite de tout vérifier à la main pour les 10 ans à venir. On ne livre pas un fragment de logiciel ; on livre de quoi travailler efficacement ensuite. C'est le critère qui départage le fini de l'inachevé : le travail fini *capitalise* — chaque pièce livrée réduit le travail futur. Le fragment de code fait l'inverse : il encaisse le bénéfice de l'annonce et laisse la charge de

l'exploitation, et le chapitre « Le code est un passif » a dit à quel taux courent ces intérêts.

Deuxième exigence : fuir ce qui réclame une maintenance sans fin. L'exemple que j'oppose régulièrement aux demandes : la liste noire de cartes graphiques — recenser les configurations défaillantes pour les désactiver d'office dans le code. Une liste noire ne sert à rien si elle n'est pas exhaustive, et pour qu'elle le reste, il faut la tenir à jour en permanence, en suivant chaque combinaison de carte, de version de pilote et de système d'exploitation : c'est une tâche qui ne peut que croître, sans que personne n'ait rien demandé. Dès qu'on s'entend dire « il faudrait une liste noire », on s'apprête à se créer un enfer : dans 6 mois, ce sera oublié ; dans deux ans, ce sera de la dette technique qui n'aide personne. Le même verdict frappe les solutions trop spécialisées et les enchaînements d'exceptions cousues main. Ce qui garantit réellement la robustesse, ce sont les tests — et l'on ne teste bien qu'un ensemble réduit de cas, ce qui reboucle sur l'éloge du simple : la fonctionnalité de niche à l'heuristique bizarroïde n'est presque jamais couverte par les tests, et le jour où elle casse, personne ne le voit venir. La maintenabilité se décide au moment d'accepter ou de refuser la fonctionnalité, pas au moment de la réparer.

Troisième exigence : ne rien lancer qu'on ne puisse mener à terme dans le temps imparti, et la mesure de cette exigence est le chantier de fondations. Remplacer une brique dont tout le reste dépend — la bibliothèque graphique qui dessine chaque bouton et chaque fenêtre — n'est pas une mise à jour : entre deux versions majeures d'une telle bibliothèque, les incompatibilités ne sont pas

des renommages de fonctions mais des changements de paradigme. Mener un tel portage à bien exige de geler tout ajout de fonctionnalité jusqu'à son terme, car continuer à fabriquer des modules et des interfaces pendant qu'on remplace le socle revient à faire creuser un trou à un ouvrier pendant que son collègue le rebouche. Darktable a entrepris de porter son interface de GTK 3 vers GTK 4 sans ce gel ; le chantier traînait depuis plus de deux ans au moment où j'écris cela, et ce n'est pas une malchance, c'est une arithmétique : chaque fonctionnalité ajoutée pendant le portage est du travail à porter en plus, produit au rythme où le portage avance[2]. Un projet sans gouvernance ne peut pas décréter un gel — personne n'y a le mandat de dire non — et c'est une des raisons pour lesquelles la question de l'organisation, qui occupe la suite de ce livre, n'est pas séparable de la technique.

Ces 3 exigences ont un dénominateur commun, qui les rend non négociables : le budget de maintenance, c'est-à-dire des humains. Les enquêtes sur les mainteneurs de logiciels libres donnent l'état du parc : environ 60 % ne sont pas payés pour ce travail, une proportion comparable a quitté son projet ou envisagé de le faire, et près d'un sur deux cite l'épuisement parmi ses raisons[3]. Voilà le budget sur lequel tirent la liste noire « qu'il suffirait de tenir à jour », le fragment livré « qu'on finira plus tard », le portage qu'on n'a pas gelé. Chaque engagement de maintenance pris à la légère est un prélèvement sur un compte déjà à découvert, et le découvert se paie en burnout — le sujet reviendra, pour lui-même, au chapitre « Le travailleur invisible ». Et l'équilibre de ce livre en dépend : soutenir qu'on doit quelque chose à l'utilisateur

oblige à la même chose envers celui qui tient l'outil debout, faute de quoi on n'aura fait que déplacer l'exploitation d'un cran.

L'objection en face est un slogan d'atelier : *ship it now, fix it later* — livre maintenant, répare plus tard ; le mieux est l'ennemi du bien ; les vrais artistes livrent. Le vrai dans le slogan est ce que le chapitre « Décider, puis tenir » a établi : il faut livrer tôt pour mesurer, et l'itération vaut mieux que la spécification infinie. Mais le slogan joue sur un glissement : livrer tôt une *solution minimale fonctionnelle* n'est pas livrer un *fragment inachevé*. Le minimum viable inclut la chaîne qui rend la chose exploitable et maintenable — le débruiteur sans sa chaîne d'administration n'est pas un produit minimal, c'est une démonstration technique sur laquelle on a vissé un manche. Et « plus tard » est un emprunt : il suppose un budget futur de réparation, tiré sur le compte qu'on vient de voir — non payé, épuisé, démissionnaire. Dans l'industrie, *fix it later* est parfois un pari finançable ; dans le logiciel communautaire, c'est presque toujours un euphémisme pour *jamais*. La version honnête du slogan existe, et ce livre la défend : livrer tôt ce qui est fini, petit mais entier — et refuser de confondre la modestie du périmètre avec l'inachèvement du contenu. La nuance est subtile, et visible seulement avec l'expérience.

Finir et maintenir ferment la boucle ouverte au chapitre « Poser le problème » : le problème posé, mesuré, tranché, construit, doit être *tenu dans la durée*, et la durée se budgète comme le reste. Mais tenir dans la durée, pour un logiciel, prend une forme sociale précise : la version stable, livrée à un public qui va y vivre.

Livrer n'est pas larguer, et c'est l'objet du chapitre suivant, qui ouvre la partie du livre consacrée au collectif.

---

1. Brooks, F. P. (1975). *The Mythical Man-Month*. Reading, MA : Addison-Wesley https://openlibrary.org/books/OL5044740M/The_mythical_man-month, chap. 1, « The Tar Pit », section « The Joys of the Craft » — les joies du métier de programmeur, suivies immédiatement de ses peines : le chapitre entier est construit sur ce contraste. ↩

2. Dépôt Darktable-org/Darktable, *issue* n° 15920 « Gtk4 Migration - Progress » https://github.com/Darktable-org/Darktable/issues/15920, ouverte le 21 décembre 2023 et toujours ouverte à la date d'écriture — soit plus de deux ans. Le travail de portage proprement dit n'a commencé à atterrir qu'à l'automne 2025 (une dizaine de *pull requests* « gtk4-prep » encore ouvertes à la mi-2026) : le chantier est donc resté déclaré, mais dormant, l'essentiel de ces deux ans. Pendant le même intervalle, le projet a livré 6 versions à fonctionnalités nouvelles sur sa cadence semestrielle (4.6 en décembre 2023, jusqu'à 5.6 en juin 2026) — la preuve directe de l'absence de gel. Une évaluation indépendante versée au dépôt en mars 2026 (*issue* n° 20433 https://github.com/Darktable-org/Darktable/issues/20433) chiffre la base à environ 506 000 lignes de C, dont quelque 205 fichiers touchant directement GTK, et qualifie la migration de « chantier de plusieurs mois, vaste et à haut risque ». ↩

3. Tidelift, *The State of the Open Source Maintainer*, enquêtes 2023 et 2024 https://www.tidelift.com/open-source-maintainer-survey-2024 : environ 60 % des mainteneurs interrogés ne sont pas payés pour ce travail ; 58 à 60 % ont quitté un projet ou envisagé de le faire ; 44 % citent le burnout (2024). La proportion de mainteneurs non payés était de 46 % en 2021 — la tendance est à la dégradation. ↩

Partie IV — Le collectif

19. Livrer, pas larguer

Une version stable est un ensemble fonctionnel achevé. Ce n'est pas l'état du logiciel arrêté à une date arbitraire. La différence entre les deux définitions paraît administrative ; elle décide en réalité de qui paie le prix de la sortie, et ce chapitre, qui ouvre la partie du livre consacrée au collectif, est entièrement contenu dans cette question. La trajectoire qu'on va chiffrer n'a rien d'unique : elle attend tout projet bénévole qui sort à date fixe, et c'est pour qu'on puisse la reconnaître à temps qu'elle est chiffrée. Voici le modèle que je combats : ouvrir tous les chantiers de front sans jamais garantir un état cohérent, puis décréter « stable » la branche principale gelée dans l'état où elle se trouve à la date convenue. La sortie n'est pas commandée par un ensemble fonctionnel achevé, mais par un calendrier ; si une fonctionnalité importante n'est pas prête, tant pis, elle viendra plus tard. Le symptôme de ce modèle a un nom dans mon expérience : le *rush* d'après-version. Un torrent de signalements sur le traqueur de bugs juste après la sortie, une version corrective bâclée deux semaines plus tard. Traditionnellement, le rush a lieu *avant* la version : on boucle, on teste, on livre, et l'on peut enfin souffler. Inventer le rush d'après, c'est prolonger l'épuisement au-delà de la ligne d'arrivée — on ne peut même pas prendre de vacances en

se disant que c'est livré. Et j'en tire la lecture qui me paraît évidente : un traqueur qui explose au moment d'une sortie « stable » signifie que la version n'était pas stable.

La discipline contraire existait, et je l'ai vue fonctionner avant de la voir abandonner. Ce sont les *versions candidates* : l'ensemble fonctionnel est figé, on annonce publiquement qu'il n'y aura plus de nouveauté pour cette version, et les candidates successives — RC0, RC1, RC2 — ne sont plus que des étapes de correction, jusqu'à ce que la dernière candidate devienne la version stable parce qu'il n'y a plus rien à corriger. Le gel fait tout le travail : il transforme la sortie en un état atteint plutôt qu'en une échéance subie. Darktable appliquait cette discipline ; elle a été abandonnée quand la gestion des versions a changé de main, au nom de l'intégration continue et des méthodes agiles, dont une lecture courante tient tout gel pour une lenteur. Le coût du renoncement est humain avant d'être technique : quelque chose casse 15 fois par jour, et les rapports partent dans tous les sens. À quoi s'ajoute une leçon sur le test, contre-intuitive et mesurée. Le testeur de bêta traditionnel est techniquement avancé : il sait rédiger un rapport exploitable. Ouvrir le test au grand public trop tôt récolte *plus* de rapports de *moins* bonne qualité — non par la faute des gens, mais parce qu'ils n'ont pas les moyens de décrire ce qu'ils ont sous les yeux. La recherche sur les rapports de bugs confirme le décalage : ce que les développeurs valorisent le plus — les étapes de reproduction, les traces d'exécution, les cas de test — est précisément ce que les rapporteurs ordinaires trouvent le plus difficile à fournir[1]. On dilue l'effort : au lieu de corriger les bugs en tandem avec ceux qui savent les décrire efficacement, on

s'épuise à décoder les rapports de ceux qui ne le savent pas. La conclusion pratique : rester plus longtemps en laboratoire, avec un panel de testeurs choisis, et laisser les compilations quotidiennes à ceux qui choisissent de vivre dangereusement, sur la branche de développement.

Sur ce sujet, la littérature scientifique semble conclure autrement — mais on va voir qu'elle étudie un autre objet. Martin Michlmayr, dans la thèse de référence du domaine, a étudié la gestion des versions dans les grands projets libres — Debian, GNOME, le noyau Linux, OpenOffice.org, X.org — et conclut que les sorties *à date fixe* y fonctionnent comme un mécanisme de coordination : dans un projet alimenté par des flux de contributeurs que personne ne contrôle, le calendrier est le seul chef d'orchestre disponible, et la qualité s'en trouve améliorée[2]. Ainsi, quand Firefox est passé en 2011 aux sorties rapides à calendrier, l'étude la plus proche d'un test contrôlé de ma thèse du « torrent de bugs » n'a trouvé aucune augmentation significative des bugs d'après-sortie — les bugs se manifestaient plus tôt, les corrections arrivaient plus vite[3]. Le torrent prédit ne s'est pas matérialisé dans les comptages — chez Firefox. Il s'est matérialisé sur mon terrain, et les deux constats ne s'annulent pas : ils indiquent que les résultats de cette littérature ne semblent pas s'appliquer aux projets structurés comme Darktable, pour des causes qui restent à explorer — la suite de ce chapitre en propose. Et la leçon vaut dans les deux sens, car c'est ici que rôde le culte du cargo : imiter les formes extérieures d'une réussite en espérant en récolter les effets. Adopter le calendrier de Mozilla sans les moyens de Mozilla, c'est copier la forme sans la fonction ;

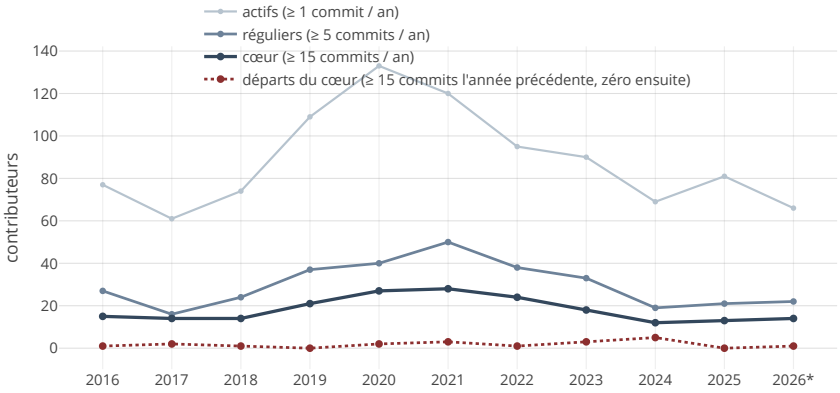
ce qui fonctionne le mieux pour certains n'est pas l'optimum pour tous.

D'abord, deux variables que ces études ne mesurent pas, et qui fondent la position. La première est le coût humain : Michlmayr mesure la coordination, Khomh compte les défauts ; aucun ne mesure la soutenabilité du travail — le rush d'après-version, l'impossibilité de clore, l'épuisement prolongé au-delà de la livraison. C'est une variable absente de leurs protocoles, pas une variable réfutée ; et le chapitre précédent a donné l'état du budget humain sur lequel elle pèse. La seconde est la capacité architecturale : une cadence de sorties sans fenêtre de gel interdit structurellement les chantiers de fondation — rien de gros ne peut jamais atterrir, puisque rien ne s'arrête jamais. Le portage GTK 4 ouvert depuis plus de deux ans, chapitre « Finir et maintenir », n'est pas une anecdote contre la sortie à date : c'en est la conséquence mécanique dans un projet qui n'a pas les moyens de Mozilla. Ensuite, la littérature elle-même a documenté les fissures de son propre récit. Sur plus de 300 000 exécutions de tests chez Firefox, Mäntylä et ses collègues ont montré que les sorties rapides *rétrécissent* le périmètre de test et *érodent* la communauté des testeurs bénévoles — Mozilla a dû embaucher des sous-traitants pour compenser — tout en corrigeant proportionnellement moins des bugs signalés[4] ; et da Costa a mesuré que, sous sorties rapides, les corrections mettent en médiane une cinquantaine de jours de plus à atteindre les utilisateurs[5]. Le récit de la sortie rapide tient donc sur les comptages de défauts, et craque sur le test bénévole, le taux de

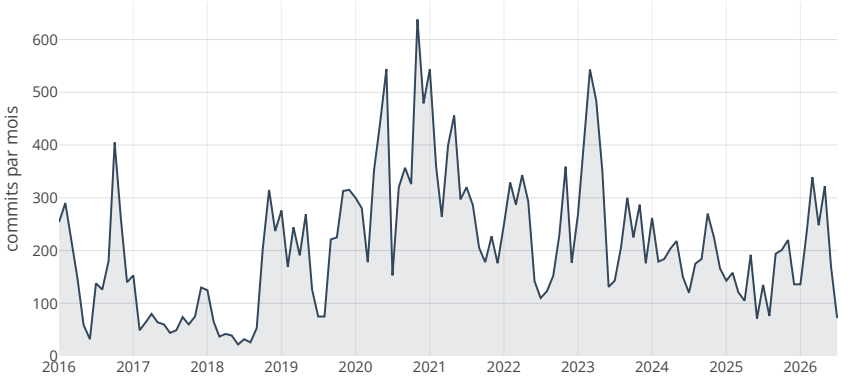
correction et la latence — c'est-à-dire très exactement sur les ressources dont vit un projet communautaire.

Cette littérature n'avait pas la contradiction sur pièces. Darktable est passé à 2 sorties à date fixe par an fin 2019, et ses registres publics permettent de suivre, sur 10 ans, ce que la cadence a fait aux 3 populations dont vit un projet : les contributeurs, les testeurs, et les utilisateurs[6].

Il faut compter d'abord les bras. Le nombre d'auteurs de commits actifs par an est passé d'un pic de 133 en 2020 à moins de la moitié aujourd'hui — et si l'on ne retient que ceux qui versent au moins 5 commits dans l'année, les réguliers sont passés de 50 en 2021 à une vingtaine depuis 2024 ; à 15 commits par an et plus, le cœur qui porte réellement le code tient en une douzaine de personnes. L'érosion qui produit cette fonte se mesure aussi : chaque année, des membres du cœur — 15 commits ou plus l'année précédente — disparaissent : 9 sur la seule période 2022-2024, dont 5 en 2024, plus du quart de l'effectif de ce niveau ; et sur 10 ans, 14 des 18 partants ne sont jamais revenus[6].



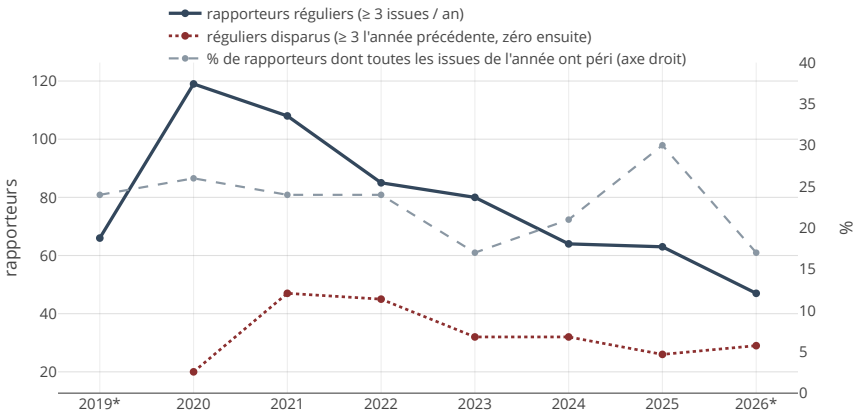
Les effectifs et leurs départs, 2016-2026 : contributeurs actifs par an et par intensité (≥ 1, ≥ 5 et ≥ 15 commits dans l'année), et départs du cœur — 15 commits ou plus une année, zéro la suivante (pointillés rouges). Commits hors fusions, auteurs dédupliqués ; 2026 est une année partielle, arrêtée au 22 juillet. Toutes les courbes culminent en 2020-2021 puis fondent de moitié ; le pic de départs de 2022-2024 (9, dont 5 en 2024) précède l'étiage.



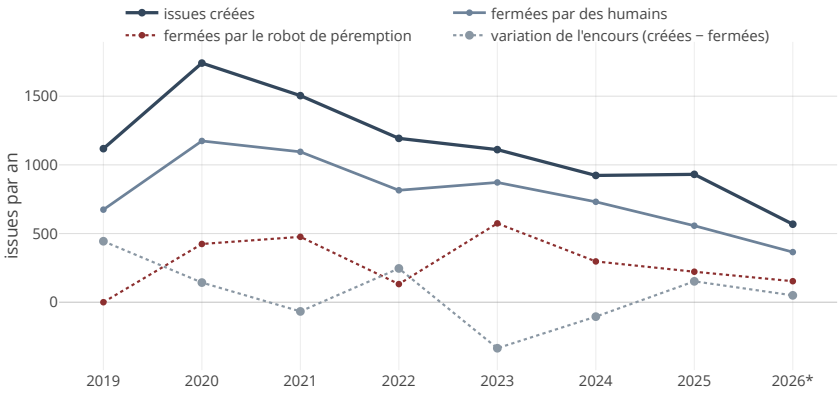
Le pouls sur 10 ans : commits par mois, 2016-2026 (hors fusions). Les crêtes sont les préparations de versions ; le sommet absolu est fin 2020 (638 commits en novembre). Depuis, l'amplitude fond : la moyenne des 12 derniers mois (196 commits) vaut à peine plus de la moitié de celle de 2020-2021 (337).

On peut d'ailleurs suivre les testeurs à la trace, et le résultat rejoint la littérature ci-dessus avec une précision embarrassante. Les rapporteurs réguliers — 3 *issues* ou plus dans l'année — étaient 119 en 2020 ; ils sont 63 en 2025, et le tourniquet tourne à plein régime : chaque année, 30 à 45 d'entre eux disparaissent, de l'ordre de 40 % de l'effectif, remplacés par des nouveaux qui ne dureront pas davantage. Et voici la mesure qui explique pourquoi ils ne durent pas : un rapporteur sur 4, bon an mal an, voit la *totalité* des problèmes qu'il a signalés dans l'année finir fermée automatiquement par le robot qui nettoie les rapports de bugs inactifs — pour un quart des testeurs, la réponse du projet à l'intégralité de leur contribution annuelle aura été le silence, puis

l'automate ; la cohorte 2025 en est déjà à 30 %. Personne ne teste longtemps pour un projet qui ne répond pas, et la qualité des rapports meurt avec la relation : la part des *issues* qui ne reçoivent jamais le moindre commentaire a doublé depuis 2019[7].



Les testeurs qui s'en vont : rapporteurs réguliers (≥ 3 issues dans l'année), réguliers disparus l'année suivante, et part des rapporteurs dont toutes les issues de l'année ont fini périmées par le robot (axe droit). Le traqueur GitHub naît en 2019 (l'échantillon 2019 couvre 90 % de l'année, plafond de l'API) ; 2026 est partiel. Pic de testeurs en 2020, moitié perdue depuis, un renouvellement d'environ 40 % par an.



Le déclin de la capacité de fermeture, 2019-2026 : issues créées, fermées par des humains, fermées par le robot de péremption. Les fermetures humaines chutent de 53 % entre 2020 et 2025 — au rythme exact de la fonte du cœur travaillant (−52 %) — pendant que le flux entrant ne recule que de 47 % : l'écart est comblé par le robot et par l'encours. La ligne pointillée donne la variation annuelle de l'encours (créées moins fermées, robot compris) : positive, le retard grossit ; négative, il se purge — et il ne se purge en masse que par le robot (2023). Depuis 2025, il regonfle ; cumul depuis 2019 : environ 530 issues de retard supplémentaires.

La série porte d'ailleurs une cicatrice datée, et je peux la signer : 2022. Les fermetures humaines d'*issues* y plongent d'un quart en un an — de 1 095 à 815, la chute la plus brutale de toute la série — alors que l'effectif, cette année-là, n'avait encore que faiblement baissé. C'est l'année dont j'ai témoigné qu'elle fut la première où je ne parvenais presque plus à faire atterrir une correction, alors que je totalisais 4 ans d'expérience à temps plein sur le projet :

tous les bugs que je lisais pointaient vers des comportements hautement aléatoires, gérés par des fichiers de code source devenus incompréhensibles. Mes commits annuels passent d'environ 200–280 entre 2019 et 2021, à moins de 80 en 2022 ; ceux qui mentionnent une correction, d'environ 50 par an à 19[8].

Le rush d'après-version lui-même se mesure. Autour de chaque version majeure, j'ai compté les commits de correction dans les 30 jours qui précèdent et les 30 qui suivent la sortie. Le verdict a deux époques :

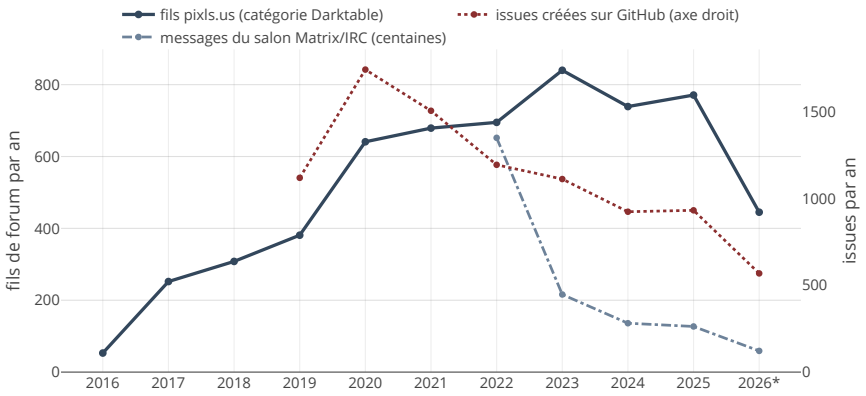
- **Jusqu'à la mi-2021**, l'après dépasse l'avant de 27 % : le torrent que je décrivais existait bel et bien. La 3.4 de décembre 2020 en est le sommet — 98 corrections le mois d'avant, 151 le mois d'après.
- **Depuis la 3.8 de décembre 2021**, le rapport s'inverse : les corrections d'après-sortie sont inférieures d'un tiers à celles d'avant. Mais…

L'inversion n'est pas un assainissement — l'encours d'*issues* et les latences disent le contraire. La capacité de se ruer sur les bugs a simplement disparu avec l'effectif : un rush suppose des bras, et ils sont partis. Le rush d'après-version n'a pas été résolu ; il a été déplacé à plus tard dans la file d'attente[9].

Le déclin simultané des ouvertures et des fermetures d'*issues* pourrait s'expliquer autrement. Première hypothèse : les *issues* d'autrefois auraient été gonflées de demandes de fonctionnalités, traitées depuis, et amenant à un logiciel à présent complet — les étiquettes la démentent : les demandes étiquetées comme telles

n'ont jamais dépassé quelques dizaines par an, stables, sur un millier d'entrées. Deuxième hypothèse : les utilisateurs signaleraient moins parce que tout va mieux — le sondage des rapporteurs dit l'inverse, et il dit qui est parti : sur un échantillon constant de 300 *issues* par an, le nombre de rapporteurs distincts *monte* (129 en 2019, 215 en 2025) pendant que le nombre d'*issues* par rapporteur *chute* (2,3 à 1,4) — les rapporteurs assidus, ceux qui suivaient leurs signalements et en déposaient beaucoup, ont disparu ; reste le rapport isolé du passant, aux trois quarts un primo-déposant, chaque année. Ce sont les plus investis qui se sont tus — le retrait de la voix, mesuré au guichet. Troisième hypothèse : la capacité de fermeture baisserait plus vite que le flux — exact : les fermetures humaines d'*issues* (hors robot) ont fondu de 53 % entre 2020 et 2025, au rythme précis du cœur travaillant (−52 %), pendant que le flux entrant ne baissait que de 47 % ; l'écart est comblé par le robot, dont la part se maintient, et par l'encours, qui regonfle. Et il faut fermer la porte à une lecture rassurante : ces deux baisses ne sont pas parallèles. Rapportées au personnel, les fermetures par tête sont constantes — 43 par personne et par an en 2020, 43 en 2025 : les restants n'ont rien absorbé de la perte, la capacité suit l'effectif, exactement. Rapportées au flux, elles reculent : les humains fermaient l'équivalent de 67 % du flux entrant en 2020, 60 % en 2025 — si la baisse des signalements était un soulagement, ce taux serait monté ; il a baissé. Une équipe réduite de moitié, au rendement individuel inchangé, ne tient même plus un flux réduit de moitié. Moins d'*issues* parce que ceux qui y croyaient sont partis ; moins de fermetures parce que ceux qui fermaient sont partis ; et la différence pourrit ou se fait balayer par l'automate[10].

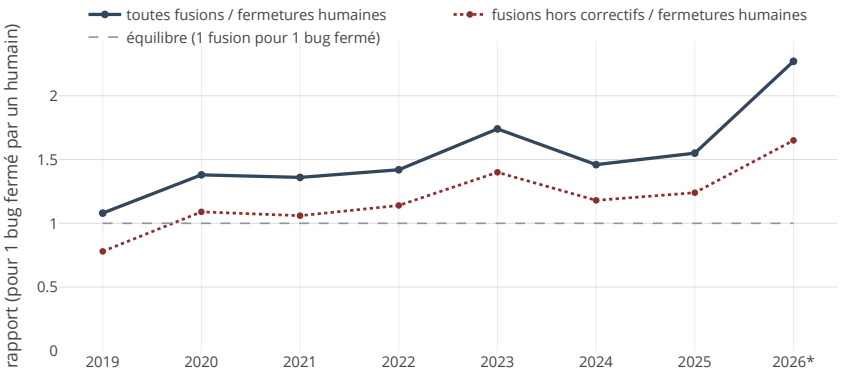
Une contre-épreuve extérieure achève ce test. Si les utilisateurs signalaient moins parce qu'ils sont moins nombreux qu'avant, ou moins engagés, leur forum le montrerait. C'est l'inverse : sur discuss.pixls.us, la catégorie Darktable n'a jamais été aussi vivante — 641 fils ouverts en 2020, 840 au sommet de 2023, 771 en 2025, et 2026 tient le même rythme ; les messages y atteignent des records. Pendant que les *issues* fondaient de moitié, la conversation des utilisateurs ne fléchissait pas d'un fil, et les nouveautés continuaient d'affluer dans le logiciel, ce qui paraît incompatible avec une stabilisation. Le déclin n'est donc pas celui de la communauté : c'est celui du canal qui la relie au développement. Les utilisateurs n'ont pas quitté le logiciel — ils ont cessé de parler à ses développeurs[11]. Et un troisième lieu confirme de quel côté est le déclin : le salon de discussion du projet lui-même — le canal Matrix/IRC où développeurs et habitués se parlent — est passé de 65 000 messages en 2022 à un rythme d'environ 10 000 par an en 2026, une fonte de 85 % ; au point qu'un des principaux « interlocuteurs » du salon est aujourd'hui l'automate qui annonce les commits[12]. Le forum des utilisateurs vit ; le traqueur et le salon des développeurs se vident : ce n'est pas le public qui manque, c'est la maison qui ne reçoit plus.



Le découplage, sur 3 lieux : fils de discussion ouverts dans la catégorie Darktable de discuss.pixls.us (axe gauche) contre issues créées sur le traqueur GitHub (axe droit) et messages du salon Matrix/IRC du projet (en centaines, axe gauche). La communauté d'utilisateurs ne décline pas — le canal qui la relie au développement et la conversation des développeurs eux-mêmes, si. Le traqueur GitHub naît en 2019, le journal du salon commence en février 2022 (sa première année est donc déjà tronquée de 6 semaines) ; 2026 est partiel.

Enfin, le flux de nouveautés n'a pas ralenti au même rythme que la réparation des bugs. La composition du flux fusionné de propositions complète le tableau : la part des ajouts de code y est restée stable, autour de 20 %, pendant que les fermetures humaines de bugs s'effondraient — le rapport entre nouveautés fusionnées et bugs fermés par des humains est ainsi passé d'environ un pour un en 2020-2021 à 1,5 pour un en 2026. Le projet fusionne de la nouveauté plus vite qu'il ne répare

l'existant, et l'écart s'élargit — en comptant toutes les fusions, correctifs compris, le rapport atteint 2,3 pour un sur 2026[7].



Fusionner plus vite qu'on ne répare : nombre de propositions de code fusionnées pour chaque bug fermé par un humain (robot exclu), en comptant toutes les fusions puis les seules fusions non correctives (« fix » absent du titre). Sous la ligne d'équilibre, le projet répare plus qu'il n'ajoute ; au-dessus, il s'endette. 2026 est arrêté au 22 juillet.

Reste à dire ce que ce rythme fait aux gens — car les registres mesurent des volumes, pas de la fatigue. Une nouveauté fusionnée a deux destinataires. Pour le développeur, elle devient la charge d'entretien de l'année suivante — le chapitre « L'ambition non héroïque » chiffre ce que coûte une fonctionnalité après sa livraison. Pour l'utilisateur investi, celui qui connaît l'outil et s'en sert pour travailler, elle défait une part de ce qu'il avait appris. Les deux destinataires tirent donc sur la même corde, par des mécanismes opposés : l'un entretient ce

qu'il n'a pas demandé, l'autre désapprend ce qu'il savait. Si l'hypothèse tient, elle explique ce que le seul déclin des effectifs n'explique pas — pourquoi ce sont les plus investis qui partent les premiers, et des deux côtés à la fois. Ce que l'utilisateur perd dans l'opération a d'ailleurs un nom comptable : des années d'apprentissage, une bibliothèque de réglages, un flux de travail — un capital qu'il a constitué en faisant confiance. Le détruire par inadvertance est le premier des manquements, et le plus difficile à réparer.

La littérature avait écrit ce scénario avant les données. Mäntylä, cité plus haut, l'avait mesuré : les sorties à cadence fixe rétrécissent le périmètre de test et *érodent la communauté des testeurs bénévoles* — Darktable est passé aux sorties à date fixe fin 2019, le pic de testeurs est en 2020, la moitié a disparu depuis : c'est la trajectoire mesurée chez Firefox, sans les sous-traitants que Mozilla embaucha pour compenser. Bettenburg, cité plus haut, avait montré que le rapport de bug vit du dialogue entre rapporteur et développeur ; le silence d'un quart des cohortes étouffe ce dialogue à la racine. Hirschman, au chapitre « Le cercle fermé », explique l'ordre des départs — les plus investis, les plus exigeants sur la qualité, sortent les premiers : ce sont exactement les rapporteurs assidus du sondage ci-dessus. Et l'on tient, pour finir, la version mesurée d'un dogme que le chapitre « Penser contre soi » démontera : la loi de Linus promettait qu'avec assez d'yeux, tout bug est superficiel. Le registre tient le compte des yeux : il fond.

La résolution du problème est un partage des régimes, et elle borne ma propre thèse en même temps que la leur. La liste des

projets étudiés par la littérature n'est pas homogène, et porte sur des projets, certes libres, mais soutenus par des entreprises et fondations employant des salariés. Un seul de ses membres est un logiciel de production comparable à un logiciel de retouche (la suite bureautique OpenOffice). Debian ne conçoit pas de logiciel au sens de ce livre : une distribution assemble des pièces produites ailleurs et résout des ensembles de dépendances — un travail d'intégration, pas de conception. Le noyau Linux et X.org, le serveur d'affichage, sont des infrastructures sans interface utilisateur propre : ils se testent machine contre machine, sans panel ni habitus. GNOME conçoit bien un environnement de bureau, mais adossé aux salariés des grands distributeurs commerciaux de Linux. Mozilla Firefox est un logiciel de consommation de contenu (navigateur internet), pas de création. L'argument de coordination de Michlmayr vaut donc pour ce qu'il a étudié : des flux de contributeurs massifs et incontrôlés, sur des objets qui s'intègrent ou se testent mécaniquement, avec des salariés au centre. Mais une petite équipe bénévole qui *conçoit* une application à interface, contrôle son périmètre et peut geler ce qu'elle a promis, n'est dans le champ d'aucune de ces études — ni par la nature de l'objet, ni par l'échelle des effectifs, ni par le statut des travailleurs ; pour elle, la sortie par ensemble fonctionnel, tenue par la discipline des candidates, reste défendable — et c'est là que je travaille. L'erreur symétrique serait d'importer le calendrier de la cathédrale dans l'atelier, ou la discipline de l'atelier dans la cathédrale. Ma position n'est pas une thèse rivale : c'est le cas que ces études n'ont pas regardé — et c'est le cas ordinaire, car un projet libre a bien plus de chances de traverser une phase Darktable qu'une phase Mozilla ; les

géants salariés sont l'exception de l'écosystème, pas sa règle. Le cas Darktable montre que les conclusions tirées de Debian ou de Firefox ne se généralisent pas à tout le logiciel libre, et le problème est en amont, dans l'idée même de comparer des projets qui n'ont à peu près rien en commun, sinon la licence. « Logiciel libre » est une catégorie juridique dans le droit de la propriété industrielle et intellectuelle, pas une catégorie d'organisation ; tant qu'on échantillonnera dedans comme dans un objet homogène, les résultats vaudront pour les projets qui ressemblent aux échantillons, et pour eux seulement.

Livrer, pas larguer : livrer un ensemble achevé à des gens qui vont y vivre, au lieu de larguer par-dessus bord ce que le calendrier a trouvé sur la branche. La question qui suit est celle du sujet de la phrase : *qui* décide de ce qui est achevé, qui gèle, qui tient le gel ? On voudrait répondre : le collectif. Il faut regarder ce que les collectifs savent réellement décider, et c'est l'objet du chapitre suivant : le comité n'inventera rien.

---

1. Bettenburg, N., Just, S., Schröter, A., Weiss, C., Premraj, R. & Zimmermann, T. (2008). « What Makes a Good Bug Report ? ». *Proceedings of FSE 2008*, ACM https://doi.org/10.1145/1453101.1453146 — enquête auprès de 466 développeurs d'Apache, Eclipse et Mozilla : étapes de reproduction, traces et cas de test sont les éléments les plus valorisés et les plus difficiles à obtenir des rapporteurs (« information mismatch »). ↩

2. Michlmayr, M. (2007). *Quality Improvement in Volunteer Free and Open Source Software Projects : Exploring the Impact of Release Management*. Thèse de doctorat, Université de Cambridge — études de cas : Debian, GNOME, Linux, OpenOffice.org, Plone, X.org https://www.cyrius.com/publications/michlmayr-phd/ . Voir aussi Michlmayr, M., Fitzgerald, B. & Stol, K.-J. (2015). « Why and How Should Open Source Projects Adopt Time-Based Releases ? ». *IEEE Software*, 32(2), 55-63 https://doi.org/10.1109/MS.2015.55 . ↩

3. Khomh, F., Dhaliwal, T., Zou, Y. & Adams, B. (2012). « Do faster releases improve software quality ? An empirical case study of Mozilla Firefox ». *Proceedings of MSR 2012*, IEEE https://doi.org/10.1109/MSR.2012.6224279 ; version étendue : *Empirical Software Engineering*, 20(2), 2015 https://doi.org/10.1007/s10664-014-9308-x . Pas d'augmentation significative des défauts d'après-sortie au passage de Firefox aux sorties rapides ; les défauts se manifestent plus tôt, les corrections sont plus rapides. ↩

4. Mäntylä, M. V., Adams, B., Khomh, F., Engström, E. & Petersen, K. (2015). « On rapid releases and software testing : a case study and a semi-systematic literature review ». *Empirical Software Engineering*, 20(5), 1384-1425 https://doi.org/10.1007/s10664-014-9338-4 — 312 502 exécutions de tests, Firefox 2006-2012 : périmètre de test rétréci, communauté de testeurs bénévoles érodée (recours à des sous-traitants), proportion moindre de bugs signalés corrigés. ↩

5. da Costa, D. A., McIntosh, S., Kulesza, U. & Hassan, A. E. (2016). « The Impact of Switching to a Rapid Release Cycle on the Integration Delay of Addressed Issues ». *Proceedings of MSR 2016*, ACM https://doi.org/10.1145/2901739.2901764 — sous sorties rapides, les corrections intégrées mettent en médiane environ 50 jours de plus à parvenir aux utilisateurs. ↩

6. Données arrêtées au 22 juillet 2026 ; dépôt https://github.com/Darktable-org/Darktable (API GitHub et historique git complet, commits hors fusions, auteurs dédupliqués par nom ; le détail du flux quotidien et de la péremption automatisée est au chapitre « Le comité n'inventera rien »). Effectifs 2016-2026 : ≥ 1 commit/an : 77, 61, 74, 109, 133, 120, 95, 90, 69, 81, 66* ; ≥ 5 : 27, 16, 24, 37, 40, 50, 38, 33, 19, 21, 22* ; ≥ 15 : 15, 14, 14, 21, 27, 28, 24, 18, 12, 13, 14* (2026 partiel). Départs du cœur (≥ 15 commits l'année n−1, zéro l'année n) : 1, 2, 1, 0, 2, 3, 1, 3, 5, 0, 1* ; 4 des 18 partants de 2016-2025 ont recommis un jour. Pouls mensuel : n = 26 865 commits, 2016-2026. ↩ ↩

7. Balayage intégral des *issues* créées de 2019 au 22 juillet 2026 (n = 8 971 ; l'API plafonne l'échantillon 2019 à 1 000 sur 1 118, soit 90 % de l'année). Les bornes ne sont pas choisies : Darktable a migré son suivi de bugs d'un Redmine auto-hébergé vers GitHub fin 2018 — 13 *issues* GitHub en 2018, 1 118 en 2019 — et l'ancien Redmine, resté consultable en 2019, redirige vers le site du projet depuis décembre 2019 (archives Wayback de redmine.Darktable.org : dernière capture servie en novembre 2019

https://web.archive.org/web/20191117225329/https://redmine.Darktable.org/). Il n'existe donc pas d'archive GitHub antérieure à 2019, et les registres Redmine n'ont pas été repris. Rapporteurs réguliers (≥ 3 *issues*/an) : 66*, 119, 108, 85, 80, 64, 63, 47* ; disparus (≥ 3 l'année n−1, zéro l'année n) : 20, 47, 45, 32, 32, 26, 29 (2020 ⟶ 2026) ; part des rapporteurs dont toutes les *issues* de l'année portent l'étiquette de péremption : 24, 26, 24, 24, 17, 21, 30, 17 % (les cohortes jeunes n'ont pas fini de vieillir) ; *issues* sans aucun commentaire : 3,4 % (2019) ⟶ 6,0 % (2025) et 12,5 % sur 2026 en cours ; en revanche, sur un échantillon de 68 *issues* réparties sur 2019-2025, 93 % ont reçu au moins une première réponse humaine — c'est le suivi qui meurt, pas le premier contact. Propositions fusionnées portant « fix » dans leur titre (heuristique de titre) : 28, 21, 22, 20, 20, 19, 20, 28 % (2019 ⟶ 2026) ; rapport (fusions non correctives) / (fermetures humaines) : 0,8 (2019) ; 1,1 ; 1,1 ; 1,1 ; 1,4 ; 1,2 ; 1,2 ; 1,65 (2026) ; rapport (toutes fusions) / (fermetures humaines) : 1,1 ; 1,4 ; 1,4 ; 1,4 ; 1,7 ; 1,5 ; 1,6 ; 2,3. ↩ ↩

8. Fermetures humaines d'*issues* 2021, 2022, 2023 : 1 095, 815, 872 ; fermetures du robot : 476, 132, 574 ; cœur à 15 commits et plus : 28, 24, 18. Mes commits dans le dépôt Darktable (hors fusions) : 197 en 2019, 281 en 2020, 210 en 2021, puis 66 à 78 en 2022 selon la méthode d'appariement des alias ; ceux dont le message mentionne une correction (« fix ») : 76, 57, 46, puis 19. Le chapitre « Le cercle fermé » date et documente le conflit. ↩

9. Fenêtres de 30 jours autour de chaque version X.Y.0 (étiquettes du dépôt), commits hors fusions dont le message mentionne une correction, totaux dédoublonnés par date de sortie : 433 avant contre 548 après de la 2.2 (2016) à la 3.6 (juin 2021), soit +27 % ; 413 avant contre 273 après de la 3.8 (décembre 2021) à la 5.6 (juin 2026), soit −34 %. Première version corrective à 8-84 jours de la sortie (médiane ≈ 55). ↩

10. Tests du 22 juillet 2026. Demandes de fonctionnalités : étiquette « feature: new » sur les *issues* créées par an, 2019-2025 : 9, 11, 38, 18, 35, 29, 22 — jamais plus de 4 % du flux (limite : 44 % des *issues* récentes ne sont pas étiquetées ; un espace de discussions GitHub existe par ailleurs). Rapporteurs : échantillon constant des 300 premières *issues* de chaque année, 2019-2025 ; rapporteurs distincts : 129, 165, 160, 183, 172, 203, 215 ; *issues* par rapporteur : 2,33, 1,82, 1,88, 1,64, 1,74, 1,48, 1,40 ; part de primo-déposants stable entre 72 et 83 %. Capacité : fermetures humaines (total moins robot) 1 174 en 2020 ⟶ 557 en 2025 (−53 %) ; cœur à 15 commits et plus : 27 ⟶ 13 (−52 %) ; *issues* créées : 1 741 ⟶ 931 (−47 %). Dérivés : fermetures humaines par membre du cœur, 43,5 (2020) et 42,8 (2025) — constantes à 2 % près ; couverture (fermetures humaines rapportées aux *issues* créées la même année) : 67 % (2020) ⟶ 60 % (2025) ; les fermetures de 2025 représentent 32 % du flux entrant de 2020. La baisse de couverture n'est pas un artefact d'échantillonnage, et le test se refait de tête. Question posée : si la propension à fermer n'avait pas changé et que seul le flux avait baissé, quelle serait la probabilité d'observer un écart de couverture aussi

grand par simple hasard ? En 2020, 1 174 fermetures humaines pour 1 741 créations : 67,4 % ; en 2025, 557 pour 931 : 59,8 % — un écart de 7,6 points. Sous l'hypothèse d'une propension commune (64,8 %, en groupant les deux années), la fluctuation que le hasard seul produit sur des effectifs de cette taille — l'erreur-type de l'écart — est de 1,9 point. L'écart observé vaut donc z = 3,9 erreurs-types, et la probabilité d'un écart au moins aussi grand par hasard, dans un sens ou dans l'autre (test bilatéral sur deux proportions), est $p \approx 9 \times 10^{-5}$ — moins d'une chance sur dix mille. La chute des fermetures n'est donc pas expliquée par la seule chute des signalements. Approximation assumée : la couverture rapporte les fermetures d'une année aux créations de la même année, alors qu'une fermeture peut viser une *issue* plus ancienne ; le test traite ce ratio comme une proportion. ↩

11. Comptage intégral des fils de la catégorie Darktable du https://discuss.pixls.us/c/software/Darktable/19 (API Discourse, 5 804 fils au 23 juillet 2026), par année de création : 53, 252, 308, 381, 641, 679, 695, 840, 739, 771 puis 445 pour 2016 ⟶ 2026 (partiel — soit un rythme annuel d'environ 800). Réserves : la catégorie mêle entraide, expositions et discussion générale ; les messages, comptés par année de création du fil, s'accumulent avec le temps. La comparaison porte sur les tendances : forum stable ou croissant, traqueur en déclin de moitié. ↩

12. Journal du salon Matrix public du projet (pont du canal IRC), export du 23 juillet 2026, 118 996 messages — provenance

détaillée au chapitre « Le cercle fermé ». Messages par an : 65 151 (2022, à partir du 16 février), 21 639 (2023), 13 629 (2024), 12 669 (2025), 5 908 au 23 juillet 2026. L'année 2022 est gonflée par le conflit qui occupe la partie « La liberté et ses conditions » de ce livre — mais 2023, année déjà « calme », reste le triple de 2026, et la pente ne s'inverse jamais. Parmi les 8 premiers émetteurs de 2025-2026 figure le robot d'annonce des commits. ↩

20. Le comité n'inventera rien

Le chapitre précédent s'est terminé sur une question : qui décide de ce qui est achevé, qui gèle, qui tient le gel ? Le logiciel libre a une réponse toute prête, et elle a la beauté des réponses toutes prêtes : tout le monde. La communauté discutera, le comité tranchera, la démocratie fera le reste. J'ai passé des années à l'intérieur de cette réponse, et ce chapitre est le compte rendu de ce qu'elle produit — non pour instruire le passé, mais pour donner à tout projet qui se croit sans structure le moyen de voir celle qu'il a déjà.

On pourrait croire qu'en asseyant dix, vingt ou trente subjectivités dans la même salle, on obtienne par moyenne une sorte d'objectivité globale. On n'obtient rien de tel : on obtient la somme de dix, vingt ou trente préférences personnelles — le plus souvent, la solution que chacun a déjà pratiquée ou vue ailleurs, resservie par paresse ou pour se sentir de nouveau pertinent. Le design par comité finit toujours de la même façon : on vote pour le dessin qui a recueilli le plus de « j'aime », parce que personne ne sait comment dépasser les subjectivités individuelles pour construire une solution générale viable. Or le chapitre « Poser le problème » a décrit ce que ce dépassement exige : un problème posé, des critères écrits avant les solutions, une séance préparée et animée. Le comité ne remplace aucune de ces choses ; il les remplace toutes par un scrutin.

Il faut mesurer ce que le scrutin ne peut pas porter, même honnête, même éclairé. Supposons qu'un vote préfère le texte rouge, et qu'un autre vote, un mois plus tard, préfère le fond rouge. Écrirez-vous du texte rouge sur fond rouge ? Chacune des deux décisions a gagné son scrutin ; leur somme est illisible. Le vice n'est pas dans les votants, il est dans l'objet : on ne peut voter que sur des unités — un bouton, une couleur, un module — alors que la qualité d'une conception loge dans la cohérence globale de l'ensemble, qui n'est pas une unité mais une relation entre toutes. La cohérence ne peut pas être soumise au vote, et ce n'est pas une difficulté pratique, c'est une impossibilité logique : quelle serait la question ? « Votez-vous pour que tout aille ensemble ? » Tout le monde vote oui, et le concepteur n'en sait pas plus sur la marche à suivre. La cohérence s'obtient autrement : par un plan, tenu dans le temps, et par un protocole qui oblige à penser contre soi — le chapitre « Penser contre soi » en fait l'inventaire. Or un comité, surtout sans direction ni structure, n'a ni plan ni protocole : il a des ordres du jour.

Il faut ensuite mesurer ce que le scrutin transporte en fraude. Dans la salle, les voix ne pèsent jamais leur poids nominal : les galons invisibles — ancienneté, réputation, aplomb, heures de vol réelles ou supposées — font qu'on vote rarement sur le dessin et souvent sur le dessinateur, et la suite de ce chapitre donnera le verdict de Freeman : l'absence de structure ne supprime pas ce pouvoir, elle le rend invisible. L'instrument de mesure tient en une question : chaque membre déciderait-il la même chose seul qu'en séance ? La psychologie sociale répond depuis les années 1950 que non. Dans les expériences de conformité d'Asch,

environ un tiers des réponses suivaient l'avis unanime du groupe contre l'évidence perceptive pourtant sous les yeux des sujets, trois quarts des participants cédant au moins une fois ; et il suffisait d'un seul dissident déclaré pour que la conformité s'effondre[1]. Un comité qui converge n'est donc pas forcément un comité qui pense : c'est souvent un comité où personne n'a voulu être le dissident — la pensée de groupe de Janis, qu'on retrouvera plus bas, n'est que ce mécanisme devenu culture.

D'où une conclusion à énoncer exactement : **la démocratie sur les détails de conception n'est pas indésirable, elle est impossible**. L'objet du vote n'existe pas : le détail voté est une unité, la qualité est une relation. Ce qui se vote, le chapitre « Le mot et la chose » l'a établi, ce sont les critères et les priorités : le but, pas le trait. Et la solution n'a rien pour séduire, c'est même à cela qu'on la reconnaît — elle ne s'appelle ni liberté, ni initiative, ni créativité, ni intelligence collective ; elle s'appelle structure, méthode et protocole. Nommer le but, nommer le moyen, écrire les critères avant les solutions, puis garder la lucidité sur le protocole lui-même, qui est une pièce faillible comme les autres : un protocole dont on ne connaît pas les limites n'est pas une méthode, c'est une superstition de plus.

Au scrutin, on oppose alors l'espoir inverse : ne plus rien trancher, et laisser le nombre faire — ajouter des gens jusqu'à ce que l'intelligence collective émerge. Il faut regarder d'où cet espoir tire son autorité. « Des études montrent que le groupe est plus intelligent que l'individu » : voilà, en général, la croyance optimiste et sympathique que le logiciel libre tente de véhiculer dans ses collectifs — pas un titre, pas une condition, pas une

tâche. C'est le culte du cargo du chapitre « Le culte du cargo » appliqué à la science : on garde l'incantation, on jette le protocole. On se souvient qu'il existe des travaux ; on a oublié ce qu'ils mesuraient, sur qui, dans quelles conditions, et pendant combien de temps — et l'on confond des années de conception logicielle avec les exercices d'une heure sur lesquels ces travaux reposent. Le dossier réel existe, et il dit le contraire de ce qu'on lui fait dire.

L'intelligence collective existe : sur 699 personnes réparties en groupes de deux à 5, l'étude fondatrice sur l'intelligence collective a mis en évidence un facteur général de performance des groupes, comme il en existe un pour les individus. Mais ce facteur n'est que faiblement corrélé à l'intelligence moyenne des membres, et guère plus à celle du plus brillant[2]. Ce qui le prédit, ce sont 4 conditions :

- **la sensibilité sociale** des membres — leur aptitude à lire l'état mental d'autrui ;
- **l'équité des tours de parole** — les groupes dominés par quelques voix raisonnent moins bien ;
- **la proportion de femmes**, qui agit à travers la première ;
- et, ajoutée par une autre branche de la recherche, **l'indépendance des jugements** : dès que chacun voit les estimations des autres avant de donner la sienne, la diversité des avis s'effondre sans que la justesse collective progresse, et la confiance du groupe monte pendant que sa fiabilité stagne[3].

Relisez maintenant ces conditions face à un projet libre.

- **Sensibilité sociale** : la littérature documente l'incivilité ouverte dans les hauts lieux du milieu libre — la liste de diffusion du noyau Linux en tête —, mais j'ai fait l'expérience de milieux tout à fait polis et néanmoins parfaitement violents, d'autant plus sournois. C'est que la condition ne mesure pas la courtoisie : elle mesure l'aptitude à lire l'état d'autrui, exercée au bénéfice de la tâche et des autres. Une bienveillance de façade peut cohabiter avec son inverse exact — lire les émotions d'autrui pour les retourner contre lui — et le chapitre « Le cercle fermé » démontera ce retournement pièces en main : la politesse érigée en tribunal, la police du ton.
- **Équité de parole** : le chapitre « Penser contre soi » compte qui fusionne les contributions de code externe, et la suite de celui-ci montrera qui décide. Il n'y a pas d'équité de parole sans meneur de débat, donc sans organisation.
- **Mixité** : un septième du personnel est féminin, au mieux, selon la méthode de mesure — la conclusion du livre y revient.
- **Indépendance des jugements** : tout s'y discute en fil public séquentiel, où chacun lit les avis déjà émis avant d'écrire le sien — le dispositif d'Asch déployé à l'échelle du projet. L'opinion des personnes influentes cadre l'opinion des personnes qui ajoutent leur commentaire à la suite, les schémas d'influence sont silencieux mais réels.
- **Agrégation**, enfin, car il faut encore un mécanisme qui convertisse les jugements privés en décision : le décompte des « j'aime » n'en est pas un — il ne convertit rien, il

applaudit. Faute de cela, c'est celui qui parle le plus fort qui est entendu.

Aucune de ces conditions n'est remplie. Reste alors ce que le nombre fait tout seul, et c'est de la physique élémentaire. Ajoutez des danseurs dans un bal : vous augmentez seulement la probabilité qu'on vous marche sur les pieds. C'est de l'entropie — davantage de particules en mouvement aléatoire dans un volume clos, donc davantage de collisions — et pour gagner en efficacité en ajoutant des particules, il faut qu'elles cessent de bouger au hasard, ce qui est d'autant plus difficile qu'elles sont nombreuses. L'intelligence collective n'est donc pas la somme des intelligences individuelles : c'est une propriété conditionnelle, qui n'émerge que si l'on parvient à canaliser les possibilités individuelles dans un contexte qui y est favorable et qu'il faut construire délibérément.

Il faut maintenant remettre l'intelligence collective à sa place, et la sortir du culte du cargo. Les premières études sur l'intelligence collective ont mesuré des groupes de passage, sur des tâches d'une heure qui ne demandaient guère de compétence particulière — rien qui ressemble à des années de conception logicielle distribuée. Les résultats ne sont donc que partiellement transférables à la conception en milieu open-source. Le facteur collectif et sa prédiction par la sensibilité sociale tiennent dans des groupes qui ne collaborent que par messages écrits, sans jamais se voir — lire l'état de l'autre entre les lignes fonctionne comme le lire dans les yeux[4] — et la grande méta-analyse de 2021, sur 1 356 groupes, confirme le facteur et ordonne ses prédicteurs : le processus de collaboration d'abord, la compétence

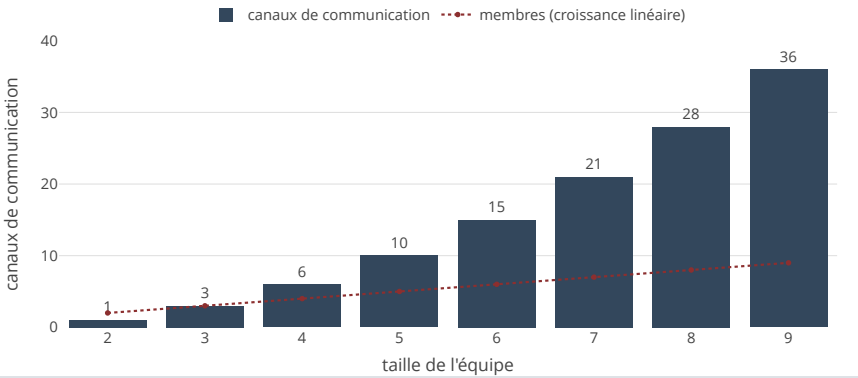
individuelle ensuite, la composition du groupe enfin[5]. Le cadre se transpose donc à nos forums libres. Mais la même méta-analyse rétablit ce que le slogan de l'intelligence collective escamote : la compétence individuelle des membres est le deuxième facteur du succès collectif, certes environ moitié moindre que le processus, mais sa part croît avec la technicité de la tâche. Ainsi, sur une épreuve de pure technique comme le sudoku, elle explique à elle seule plus de la moitié de la variance[5]. Le processus fait donc mieux raisonner un groupe compétent ; il ne crée pas la compétence. Dans le vocabulaire de ce livre : la formation et le métier ne suffisent pas à faire bien concevoir un collectif, mais sans eux rien n'émerge du tout — condition nécessaire, jamais suffisante, et donc premier poste de tout projet qui prétend concevoir. L'erreur commune des collectifs, pas seulement dans le logiciel libre — mais c'est le cas extrême où l'erreur se paie le plus cher puisque son produit est technique —, c'est de croire que le groupe puisse faire émerger la compétence ex nihilo par la seule force de la collaboration. Ça n'est pas ce que conclue la littérature.

Reste que concevoir n'est ni estimer ni deviner : c'est produire. Et sur les tâches de production d'idées, la littérature est plus dure encore pour le comité. On sait depuis les années 1980 que les groupes de *brainstorming* produisent moins d'idées, et pas de meilleures, que les mêmes individus travaillant séparément dont on met ensuite les listes en commun ; le mécanisme dominant est le blocage de production — une seule personne parle à la fois, les autres attendent, et ce qu'ils attendaient de pouvoir dire s'évapore[6]. La méta-analyse confirme la perte et lui donne une

pente : elle croît avec la taille du groupe[7]. Le titre de ce chapitre n'est pas une provocation, c'est un résultat expérimental. Et pour le cas précis de ce livre — le logiciel de bureau destiné à des utilisateurs finaux —, l'étude de terrain canonique du design logiciel dit où loge la ressource rare : sur 17 grands projets, le premier des 3 problèmes identifiés est la rareté de la connaissance du domaine d'application, et le sort des projets tenait à un ou deux concepteurs d'exception, capables de construire le pont entre le comportement attendu de l'application et les structures logicielles qui l'implémentent — souvent, de leur propre aveu, des programmeurs médiocres, mais les gardiens de la vision du projet[8]. Concevoir un outil pour photographes exige de savoir ce qu'est la photographie. Cette connaissance-là ne s'agrège pas par scrutin, ne s'improvise pas en fil de discussion, et aucun processus de groupe ne la fait émerger chez ceux qui ne l'ont pas : la partie « La discipline » l'a établi en méthode, le terrain le confirme en données.

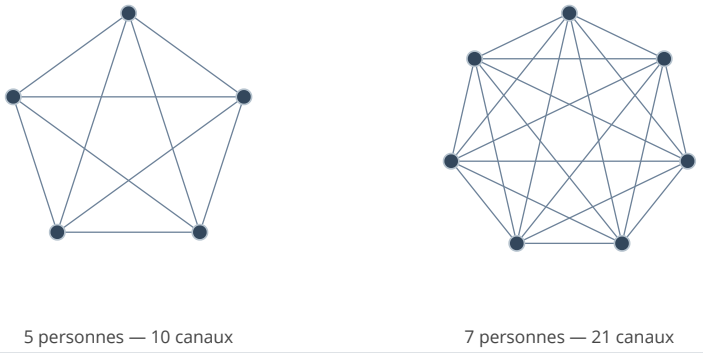
L'arithmétique de Brooks donne la pente : les canaux de communication croissent comme $n(n-1)/2$ — 6 canaux à 4 personnes, 28 à 8[9]. La psychologie des groupes donne le seuil : la performance des équipes de travail plafonne entre 4 et 7 membres ; à partir de 8, la productivité chute, et au-delà de 9 apparaissent les clans, la politique interne et la paresse sociale — l'effort individuel qui se dilue dans le nombre[10]. Et Conway a donné la conséquence architecturale, qui explique une bonne part des monstres de la partie « La machine » : un système copie la structure de communication de l'organisation qui le construit[11]. Une nuée d'individus déconnectés, sans division du travail ni

communication préalable, produit une architecture chaotique : incohérente, redondante, juxtaposée. J'en ai vu le coût à l'état pur, en nettoyant le code hérité de Darktable dans Ansel : plusieurs fonctionnalités avaient été ré-implémentées plusieurs fois, à différents moments (et parfois à 6 mois d'intervalle) et de manière incohérente, si bien que la même tâche donnait un résultat différent à différents endroits du logiciel. Deux personnes avaient résolu le même problème sans se parler, et le mainteneur n'avait pas vu la redondance ; la loi de Conway n'est pas une abstraction, c'est ce gâchis-là, répété.



La pente de Brooks : le nombre de canaux de communication, n(n−1)/2, en fonction de la taille de l'équipe (barres), contre la croissance linéaire de l'effectif (pointillé rouge). Passer de 4 à 8 personnes double l'équipe mais plus que quadruple les canaux — de 6 à 28 ; à 9 membres, chaque arrivant crée 8 conversations de plus à entretenir.

Il faut pousser cette idée jusqu'à sa forme d'ingénieur : la communication d'une équipe est un réseau, et ce réseau est un objet de conception. Dessinez le graphe — les personnes en nœuds, les canaux de communication en arêtes.

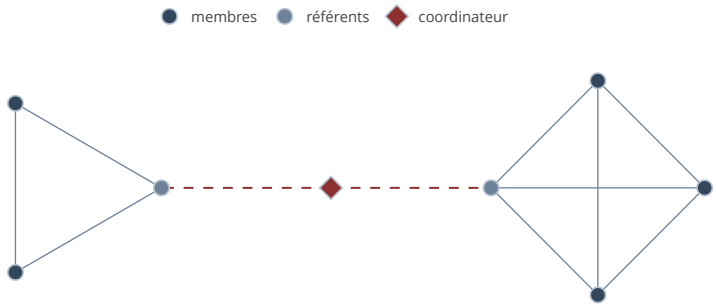


Le maillage complet, dessiné : à 5 personnes, 10 canaux — le réseau reste lisible ; à 7, 21 canaux — chaque membre tient 6 conversations, et le graphe est déjà plus dense que ce qu'aucun participant ne peut suivre. C'est la figure du bal : tout le monde parle à tout le monde, et l'information n'a plus de chemin privilégié.

La communication elle-même se dose : c'est un moyen au service d'un but, pas une fin. Depuis les années 2010, et surtout dans les milieux techniques, on a tendance à focaliser le blâme sur le manque (voire l'absence) de communication, mais trop de communication produit exactement le même résultat : on arrête d'écouter le flux épuisant d'information pour sauver un minimum de productivité. Ne pas se parler ou ne plus s'écouter aboutissent à la même rupture du flux d'information et sont les

deux écueils de la communication, qui doit être régulée, dirigée sur un sujet, avec un lieu pour les hors-sujets et un lieu pour trancher. Une fois le consensus atteint sur les objectifs, on passe à l'action : on ne communique pas pour le plaisir de communiquer.

La réponse n'est pas moins de communication, c'est une communication *structurée* : des sous-équipes de 4 à 7, dont les échanges internes sont denses et les échanges externes passent par des interfaces définies — un référent, un compte rendu écrit, un lieu de décision. On reconnaît le geste : c'est la dissimulation d'information de Parnas, appliquée à l'organisation — et ce n'est pas une analogie de mon cru, c'est la conclusion de son article de 1972, qui note que la structure des modules et celle de l'organisation finissent par se répondre. Conway se lit alors dans les deux sens : puisque le système copie le graphe de communication, on peut *concevoir le graphe de communication pour obtenir l'architecture qu'on veut* — l'industrie appelle cela la manœuvre de Conway inverse, et il existe désormais toute une ingénierie des topologies d'équipes : quelles équipes, avec quelles frontières, interagissant sur quels modes[12]. Une équipe qui laisse son réseau de communication à l'improvisation subit son architecture ; une équipe qui le dessine la choisit.



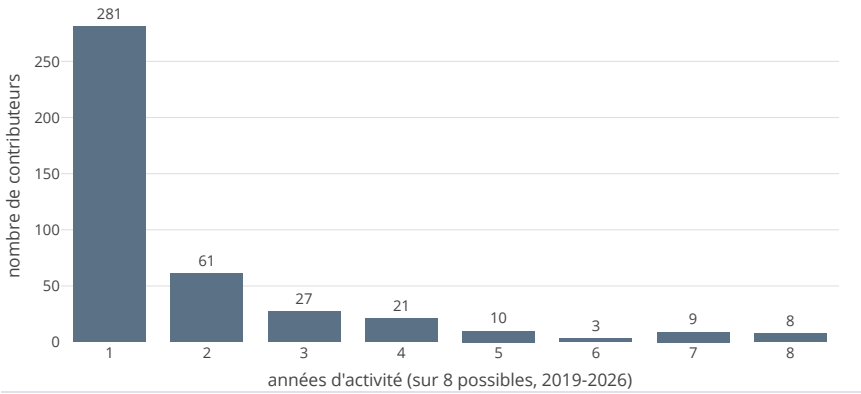
Le même réseau, dessiné au lieu d'être subi : une équipe de 3 et une équipe de 4, denses à l'intérieur, qui communiquent entre elles par leurs référents (gris) et un coordinateur (losange rouge). 8 personnes, 11 canaux — 3 dans la première équipe, 6 dans la seconde, deux d'interface — contre 28 en maillage complet : moins de la moitié des conversations, et chaque information a désormais un chemin connu. Le prix est visible aussi : le coordinateur est un point de passage obligé — un pouvoir, qui doit donc être nommé, borné et révocable, faute de quoi l'on retombe sur Freeman.

À quoi ressemble un graphe laissé à l'improvisation ? J'ai fini par le mesurer, et il faut donner les chiffres, car ils dépassent l'intuition. Sur le traqueur GitHub de Darktable, il arrive en moyenne, jour après jour et sans un seul jour de pause :

- 6 à 7 nouveau fils de discussion — *issues* (le terme du traqueur recouvre rapports de bugs et demandes de fonctionnalités indifférenciées) et propositions de code confondues — environ 2 400 par an, un volume stable sur les 3 dernières années ;

- une vingtaine de messages de discussion (mises à jour) ;
- 3 ou 4 commentaires de relecture ligne à ligne sur des propositions de code ;
- 6 clôtures ou fusions de propositions de code.

Près de 40 événements par jour, soit un toutes les 25 minutes de veille, 365 jours par an — et c'est un plancher, qui ne compte ni les revues formelles, ni les poussées de code sur les fils suivis, ni les mentions[13]. Cela peut sembler peu en valeur absolue, mais il faut comparer cela avec le personnel qui traite le flux pour en juger. Tout cela arrive sur un canal unique, adressé à tout le monde — c'est-à-dire à personne en particulier : aucun tri à l'entrée, aucun lieu séparé pour les hors-sujets, aucune interface entre l'utilisateur qui signale et le développeur qui code. C'est le maillage complet du paragraphe précédent, observé en production. Qui absorbe ce flux ? Sur 3 semaines de juillet 2026, 99 comptes ont écrit au moins un message — et deux personnes ont porté à elles seules le quart des réponses. Car la population qui tient l'autre bout du canal est minuscule, et elle rétrécit. Depuis 2019, 420 personnes ont versé au moins un commit au projet ; les deux tiers n'ont été actives qu'une seule année — le contributeur médian est un passant ; 30 ont tenu 5 ans ou plus, 8 ont été présentes chaque année, et 5 auteurs signent à eux seuls plus de la moitié des commits. Le chapitre « Livrer, pas larguer » a suivi ces effectifs dans le temps : pic en 2020-2021, moitié fondue depuis — le cœur qui porte le code tient aujourd'hui en une douzaine de personnes, actives essentiellement entre 17 h et 22 h.



Les mêmes 420 contributeurs, classés par nombre d'années d'activité entre 2019 et 2026 : les deux tiers ne restent qu'une année ; 8 personnes sont présentes chaque année.

Voilà le maillage complet en chiffres : un torrent d'interactions non triées, non hiérarchisées, qui converge de fait vers une poignée de permanents que rien ne protège — le chapitre « Travailler pour le travail » mesurera ce que coûte chaque interruption, et il faut l'imaginer payé 40 fois par jour par ceux-là mêmes qui écrivent le code. Un graphe de communication que personne n'a dessiné n'est pas un graphe libre : c'est un entonnoir.

Ce torrent, il faut le dire aussi, est un torrent en baisse. Le chapitre « Livrer, pas larguer » l'a mesuré : le flux entrant d'*issues* a reculé de près de moitié depuis 2020, et les rapporteurs assidus — ceux qui suivaient leurs signalements et en déposaient beaucoup — ont déserté le guichet ; il ne reste guère que le rapport isolé du passant. La communication ne déborde donc pas

parce qu'elle serait trop abondante : elle déborde alors même qu'elle se tarit. Et le tarissement a probablement la même cause que le débordement : un signalement qui pourrit sans réponse jusqu'à la clôture robotisée n'apprend qu'une chose à son auteur — ne plus signaler ; le manque de bras pour traiter le flux a asséché la source. Le projet souffre ainsi de trop peu de communication en général, et de plus de communication que ce qu'une équipe sans organisation peut soutenir en particulier. Le problème n'a jamais été le volume : c'est l'absence de tout dispositif entre le volume et les bras.

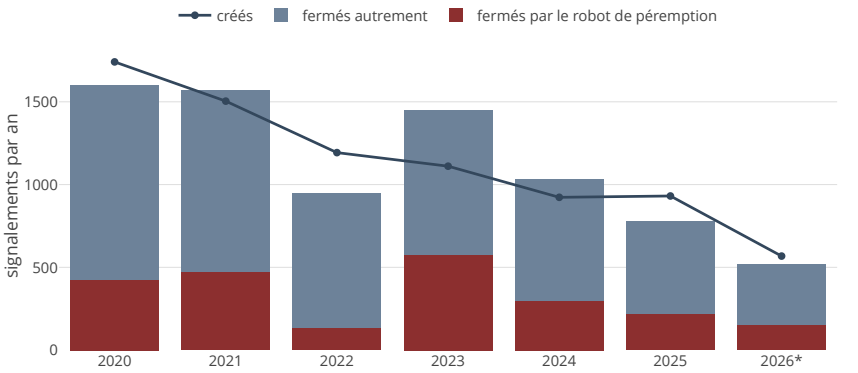
Et l'entonnoir a une conséquence qu'on n'ose pas formuler : à ce débit, le registre devient facultatif. Faute de lieux séparés — un pour signaler une panne, un pour recueillir un besoin, un pour délibérer entre contributeurs —, tout arrive au même endroit et rien n'a de rang. Ce qui n'a pas de rang se lit quand on a le temps, c'est-à-dire jamais. Le journal du salon IRC/Matrix en garde la trace, sous la forme la plus économique qui soit. En janvier 2023, un critique extérieur reproche au projet ses défauts d'interface, mesures de contraste à l'appui et mentionne l'*issue* qu'il a déposé ; personne ne l'avait vue, ce à quoi un contributeur conclut : « si tu ne lis pas les tickets, alors il n'y a pas de tickets »[14]. La phrase est une plaisanterie, et c'est ce qui la rend accablante : le mécanisme est assez familier au milieu pour faire une blague d'initiés.

Le canal principal, lui, perd l'information par construction : IRC ne conserve pas d'historique. Une conversation permanente sans archive consultable n'informe que ceux qui étaient présents au bon moment : « lire l'historique est mauvais pour la santé »,

résumait le fondateur en 2022, pendant que je soutenais depuis 4 ans que ce canal est injuste envers quiconque n'y est pas connecté en permanence[15], d'autant plus si l'on tient compte des fuseaux horaires de chacun, dans un projet international. Un projet qui délibère là où l'on ne peut pas relire ne décide pas moins : il décide sans mémoire et sans témoins. C'est la version communicationnelle du verdict de Freeman — non pas trop de structure, mais aucune, et le pouvoir qui s'y loge —, verdict que ce chapitre citera en entier une fois ses mesures posées.

Ce que l'entonnoir fait au travail se lit dans les registres car le système de versionnement Git, sur lequel se base GitHub, conserve toutes les archives, sur lesquelles on peut facilement scripter des extractions de statistiques systématisées et à l'échelle de la vie du projet. Le régime de traitement des rapports de bug est *immédiat ou jamais* : sur les 4 derniers mois, la moitié des *issues* fermées l'a été en un jour et demi médian — la réactivité est réelle — mais 47 % restent ouvertes, et le solde net grossit d'environ 170 *issues* par an ; il y a aujourd'hui plus de 500 *issues* en attente, soit une quarantaine par membre du cœur travaillant. Le tri n'existe plus : 44 % des *issues* récentes ne portent aucune étiquette, 6 % seulement ont été confirmées par reproduction — et l'étiquette la plus fréquente du registre est celle qu'appose le *robot de péremption*, l'automate chargé de constater qu'un fil est resté sans réponse. Le sort statistiquement le plus probable d'une *issue* récente, après « fermé dans la journée », est d'être déclarée morte par une machine. Et ce n'est pas un accident de l'échantillon : depuis l'ouverture du traqueur en 2019, environ 9 000 *issues* ont été déposées, et près d'une sur 3 a fini périmée par

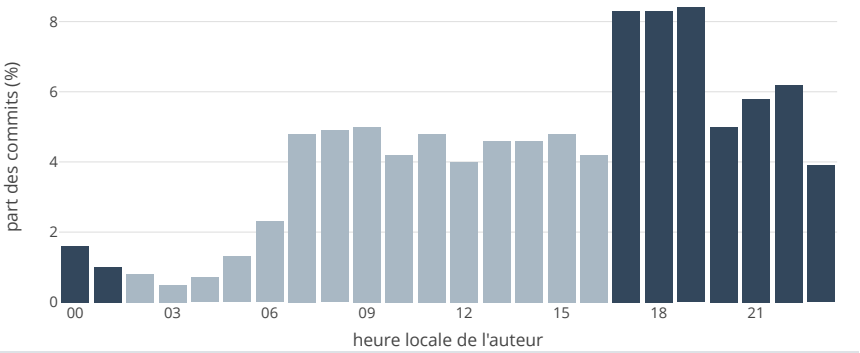
ce robot ; bon an mal an, l'automate signe 3 fermetures sur 10. Le seul acteur qui vide le registre à grande échelle n'est pas un humain qui répond : c'est un compteur d'inactivité. Et il ne faut pas croire que le dedans l'ignore : « de toute façon, personne ne regarde GitHub », écrivait un contributeur du cœur sur le salon du projet dès l'été 2022 ; et fin 2024, quand l'empaqueteur Debian du logiciel objecta qu'il préférerait « que les non-bugs soient fermés avec une explication, et que les bugs ouverts restent ouverts — l'accumulation est le moindre mal », l'opérateur du robot répondit par une question qui dit tout le régime : préféreriez-vous que ces *issues* ne soient jamais fermées, « vu leur âge et leur nombre qui grossit » ?[16] La fermeture par ancienneté et par volume n'est pas un accident déploré : c'est la politique, défendue comme telle par celui qui l'exécute.



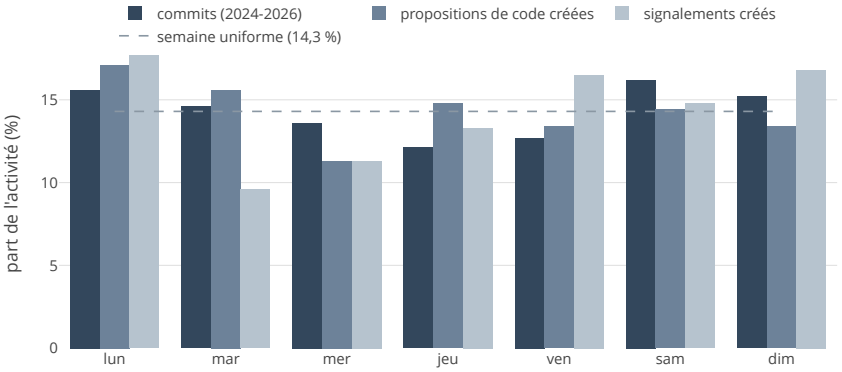
La fin de vie des issues Darktable, par année de fermeture : la part rouge est fermée par le robot de péremption — le constat automatisé qu'aucune réponse n'est venue —, la ligne donne les issues créées. Le registre ne se vide durablement qu'avec l'aide du robot ; 2026 est arrêté au 22 juillet.

Les propositions de code, elles, sont fusionnées en 10 heures médianes — vitesse qui impose une question que le chapitre « Le culte du cargo » a tranchée par l'arithmétique : à ce débit, sur ce volume, avec cette équipe — et un seul homme aux droits de fusion —, la profondeur de relecture possible est nulle. Quant au battement de cœur du projet, il n'en a pas : le flux entrant de signalements et de propositions est plat sur la semaine — les utilisateurs signalent le dimanche comme le mardi — et les commits le sont presque autant, à peine gonflés le week-end ; c'est l'heure du jour qui dit tout, car le travail se concentre entre 17 heures et minuit, heure locale de chaque auteur — 40 % des commits après 18 heures. Des gens qui ont un emploi ailleurs versent leurs soirées, tous les soirs, dans un flux qui ne connaît ni

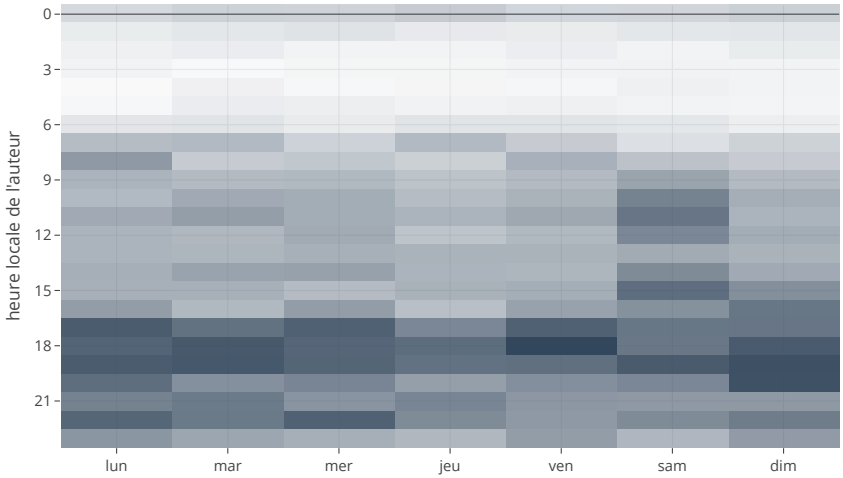
semaine ni relâche[17]. Le système tient — mais il tient en renonçant précisément à ce que ce livre appelle concevoir : trier, confirmer, prioriser, périodiser. L'hypothèse n'est pas que ces gens travaillent mal ; c'est que ce volume, sur ce graphe, est ingérable par cet effectif, et que l'organisation entière s'est reconfigurée autour de cette impossibilité, en abandonnant tout ce qui ne fait pas assez de bruit, mais aussi en faisant le vide par le découragement.



Le battement de cœur du projet : répartition horaire des 5 652 commits de 2024-2026 (hors fusions), en heure locale de chaque auteur. Le pic est en soirée — 40 % des commits après 18 h : des soirées de salariés, versées dans un flux qui ne s'arrête jamais.



La semaine qui n'existe pas : répartition par jour de la semaine des commits (2024-2026), des propositions de code et des issues créées (échantillon mars-juillet 2026). Tout reste à quelques points de la semaine uniforme : ni les utilisateurs ni les développeurs n'ont de rythme hebdomadaire — le flux ne connaît pas le repos.



La carte de chaleur du travail, 2016-2026 : 26 865 commits ventilés par jour de la semaine et heure locale de l'auteur. La tache sombre couvre les soirées de tous les jours de la semaine — le projet vit après les heures de bureau, 7 jours sur 7.

Vient alors l'argument qui tient lieu de constitution à beaucoup de projets : pas de chef, pas de hiérarchie, pas de structure — la liberté. C'est ici que se place la citation la plus importante de ce livre en matière de gouvernance, et elle vient du mouvement féministe américain, qui a fait l'expérience avant le logiciel. En 1972, Jo Freeman publiait « The Tyranny of Structurelessness » : l'absence de structure n'existe pas[18]. Un groupe qui refuse toute structure formelle ne supprime pas le pouvoir ; il le rend invisible. Des élites informelles s'y forment — celles qui étaient là avant, qui se connaissent, qui tiennent les canaux — et elles exercent un pouvoir d'autant plus irresponsable qu'officiellement il n'existe pas : on ne peut ni élire, ni évaluer, ni destituer ce qui n'a pas de

nom. Seule la structure explicite rend le pouvoir responsable, précisément parce qu'elle le rend visible. Et Freeman ajoutait le corollaire opérationnel : l'absence de structure ne peut pas diriger du travail. Tout y est. Le projet fier de n'avoir ni hiérarchie ni procédure en a pourtant une — les mêmes individus qui verrouillent les mêmes décisions pendant des années, sans mandat, sans évaluation, sans contre-pouvoir — et le prologue de ce livre a dit ce qu'il en coûte quand la pression publique est le seul levier existant. Le problème n'est pas que des gens décident ; c'est qu'ils décident sans être ni désignés ni responsables, et que le système qui les abrite rend ce verrouillage possible.

Que met-on à la place, si ni le comité ni l'absence de structure ne dirigent du travail ? Un leadership — le mot est usé, il faut donc le définir par ses 3 propriétés. Il est *gagné* : la recherche sur les communautés du libre montre que la légitimité s'y construit dans le temps et se concède conditionnellement — ni la fondation du projet ni le talent technique ne la confèrent à eux seuls ; les leaders qui durent font un travail d'organisation, pas seulement du code[19]. Il est *responsable* : structuré, visible, évaluable — exactement ce que Freeman oppose aux élites informelles ; l'autorité se borne par des mécanismes explicites, pas par la bonne volonté. Et il est *capable de dire non* : le mode de défaillance du mainteneur de Darktable est là, dans l'écart entre le code écrit et le code accepté — accepter pour être gentil du code qu'on ne pourra pas maintenir, esquiver les décisions difficiles pour préserver l'ambiance, c'est se surcharger soi-même et léguer le problème au projet. Reste le risque symétrique, car un leadership qui dit non s'entoure vite de silence : la pensée de

groupe, décrite par Janis — la cohésion qui étouffe le désaccord jusqu'au désastre[20]. Les contre-mesures s'organisent : une culture où le dissensus est un service rendu et non une attaque, des votes anonymes quand le sujet chauffe, un avocat du diable tournant ; et la recherche sur les équipes a nommé la condition de possibilité de tout cela, la sécurité psychologique — on ne signale les problèmes que là où signaler ne coûte pas[21]. Quant au vote majoritaire, il reste un outil de clôture, pas un outil de conception, et il opprime mécaniquement les minorités structurelles — le chapitre « Compter les absents » a montré qui manque dans les échantillons ; la même arithmétique vaut dans les scrutins. La ligne de partage que ce livre défendra jusqu'au bout est donc celle-ci : le collectif est souverain sur les *buts* et les budgets — ce qui mérite d'être résolu — ; la *manière* de résoudre reste un métier, avec un responsable qui a un nom.

L'objection a un texte fondateur, le plus influent de l'histoire du logiciel libre. Dans « La cathédrale et le bazar », Eric Raymond oppose au développement cathédral — planifié, hiérarchique, fermé — le bazar : chacun gratte sa démangeaison personnelle, on livre tôt et souvent, et « avec suffisamment d'yeux, tous les bugs sont superficiels »[22]. Le texte a convaincu une génération que la structure était le problème et la foule la solution. Quatre réponses. D'abord, le bazar décrit dans le texte n'a jamais été sans structure : le noyau Linux, son exemple central, a toujours eu un mainteneur au sommet et une hiérarchie de lieutenants — une cathédrale qui accepte les livraisons du bazar. Ensuite, les yeux nombreux trouvent les bugs superficiels, littéralement : l'écrasement de métadonnées du chapitre « Mesurer sans se

mentir », les 18 requêtes par survol du chapitre « Sous l'interface, les dragons » sont restés des années sous des milliers d'yeux, parce qu'une architecture ne se relit pas en passant — la profondeur n'est pas accessible au regard distribué. De plus, la démangeaison personnelle comme moteur produit exactement l'accumulation que le chapitre « L'éloge du simple » a décrite : des fonctionnalités excitantes à écrire, jamais de maintenance, ennuyeuse par définition. Enfin, le noyau Linux n'est pas une application métier avec interface graphique pour utilisateur final, mais le cœur d'un système d'exploitation qui reste invisible aux yeux des utilisateurs : combien d'utilisateurs d'Android savent qu'il est basé sur Linux ? L'interaction avec Linux est réservée aux seuls développeurs d'applications, via des fichiers de configuration et des protocoles, et aucun utilisateur « n'utilise Linux » : on utilise en fait des distributions basées sur Linux.

Deux questions séparent ici le contributeur du concepteur : mû par l'envie, le premier demande « et si on ajoutait… ? » ; mû par le problème, le second demande « qu'est-ce qui manque, à qui, et comment y remédier ? ». La première question additionne des fonctions ; seule la seconde conçoit. Mais la meilleure réponse au bazar est dans ses propres données : l'étude la plus vaste jamais menée sur le devenir des projets libres — 174 000 projets suivis — conclut que le succès se prédit par le *leadership par l'exemple*, une vision claire et des objectifs clairement énoncés[23]. Le romantisme du bazar a survécu comme slogan ; ses propres données donnent raison au modèle inverse — un projet mené par quelqu'un qui sait où il va.

Le comité n'inventera rien, l'absence de structure ne dirigera rien : dans les deux cas, le travail retombe sur quelqu'un — le même, en général, celui qui maintient, documente, trie et répond pendant que la discussion continue. Il est temps de regarder ce travailleur que tout le monde utilise et que personne ne voit. C'est l'objet du chapitre suivant.

---

1. Asch, S. E. (1951). « Effects of Group Pressure upon the Modification and Distortion of Judgments », dans H. Guetzkow (dir.), *Groups, Leadership and Men*. Pittsburgh : Carnegie Press, p. 177-190 ; exposé grand public : Asch, S. E. (1955). « Opinions and Social Pressure ». *Scientific American*, 193(5), 31-35 https://doi.org/10.1038/scientificamerican1155-31 ; monographie : *Psychological Monographs*, 70(9, n° 416), 1956 https://doi.org/10.1037/h0093718. Chiffres : environ un tiers des réponses aux essais critiques suivent la majorité unanime dans l'erreur (moins de 1 % d'erreurs en condition de contrôle) ; environ 75 % des sujets cèdent au moins une fois — et environ 25 % ne cèdent jamais : Asch lui-même insistait autant sur l'indépendance que sur la conformité. Un seul compère dissident fait tomber la conformité autour de 5 % : c'est l'unanimité qui fait le pouvoir, pas le nombre — d'où la valeur, au chapitre « Penser contre soi », du dissident institué. ↩

2. Woolley, A. W., Chabris, C. F., Pentland, A., Hashmi, N. et Malone, T. W. (2010). « Evidence for a Collective Intelligence Factor in the Performance of Human Groups ». *Science*, 330(6004), 686-688. https://doi.org/10.1126/science.1193147

Deux études, 699 participants, groupes de deux à 5 : un facteur général explique 43-44 % de la variance des performances de groupe. Corrélations avec l'intelligence des membres : r = 0,15 (moyenne) et 0,19 (maximum) — significatives mais faibles. Prédicteurs : sensibilité sociale, mesurée par le test « Reading the Mind in the Eyes » (r = 0,26) ; variance des tours de parole (r = −0,41, badges sociométriques, sur un sous-ensemble des groupes) ; proportion de femmes (r = 0,23, effet médié par la sensibilité sociale). ↩

3. Lorenz, J., Rauhut, H., Schweitzer, F. et Helbing, D. (2011). « How Social Influence Can Undermine the Wisdom of Crowd Effect ». *PNAS*, 108(22), 9020-9025 https://doi.org/10.1073/pnas.1008636108 — « social influence effect », « range reduction effect », « confidence effect » : les 3 termes sont des auteurs. La formulation canonique des conditions de la sagesse des foules — diversité des opinions, indépendance, décentralisation, et un mécanisme d'agrégation transformant les jugements privés en décision collective — est de Surowiecki, J. (2004). *The Wisdom of Crowds*. New York : Doubleday, chap. 1. https://openlibrary.org/books/OL7440777M/The_Wisdom_of_Crowds ↩

4. Engel, D., Woolley, A. W., Jing, L. X., Chabris, C. F. et Malone, T. W. (2014). « Reading the Mind in the Eyes or Reading between the Lines ? Theory of Mind Predicts Collective Intelligence Equally Well Online and Face-To-Face ». *PLoS ONE*, 9(12), e115212

https://doi.org/10.1371/journal.pone.0115212 — 68 groupes de 4, dont 36 ne collaborant que par messagerie écrite : le facteur collectif explique 41 % de la variance en ligne (49 % en présence), et le test de lecture de l'état mental prédit la performance même quand personne ne voit personne. ↩

5. Riedl, C., Kim, Y. J., Gupta, P., Malone, T. W. et Woolley, A. W. (2021). « Quantifying Collective Intelligence in Human Groups ». *PNAS*, 118(21), e2005737118 https://doi.org/10.1073/pnas.2005737118 (rectificatif : *PNAS*, 119(19), 2022, sans effet sur les conclusions) — 22 études, 1 356 groupes, 5 279 individus. Dans leurs termes : le processus de collaboration prédit l'intelligence collective davantage que la compétence des membres ; la compétence vient en deuxième, sa part dépendant de la tâche (plus de 51 % de la variance expliquée sur le sudoku) ; la proportion de femmes reste un prédicteur significatif, médié par la sensibilité sociale. ↩ ↩

6. Diehl, M. et Stroebe, W. (1987). « Productivity Loss in Brainstorming Groups : Toward the Solution of a Riddle ». *Journal of Personality and Social Psychology*, 53(3), 497-509 https://doi.org/10.1037/0022-3514.53.3.497 — sur 22 expériences antérieures recensées, 18 donnent l'avantage aux groupes nominaux et aucune aux groupes réels, en quantité comme en nombre de bonnes idées ; leur 4e expérience isole le blocage de production comme mécanisme principal, devant l'appréhension du jugement et le passager clandestin. ↩

7. Mullen, B., Johnson, C. et Salas, E. (1991). « Productivity Loss in Brainstorming Groups : A Meta-Analytic Integration ». *Basic and Applied Social Psychology*, 12(1), 3-23 https://doi.org/10.1207/s15324834basp1201_1 — perte « significative et de forte ampleur », en quantité comme en qualité, corrélée à la taille du groupe (r = 0,61 en quantité, 0,72 en qualité) ; les auteurs privilégient des mécanismes sociaux plutôt que le seul blocage — le désaccord porte sur le mécanisme, pas sur la perte. ↩

8. Curtis, B., Krasner, H. et Iscoe, N. (1988). « A Field Study of the Software Design Process for Large Systems ». *Communications of the ACM*, 31(11), 1268-1287 https://doi.org/10.1145/50087.50089 — 17 grands projets, 9 entreprises, 97 entretiens. Premier des 3 problèmes relevés : « the thin spread of application domain knowledge ». Les concepteurs d'exception, un ou deux par projet, « extrêmement familiers du domaine d'application », décrits comme ceux qui ont « sauvé » le système, se disaient souvent médiocres programmeurs et étaient reconnus comme « le cœur intellectuel du projet », gardiens de sa vision (notre traduction). ↩

9. Brooks, F. P. (1975). *The Mythical Man-Month*. Reading, MA : Addison-Wesley (éd. anniversaire 1995) https://openlibrary.org/books/OL5044740M/The_mythical_man-month — l'arithmétique des canaux d'intercommunication n(n−1)/2, et la loi qui porte son nom : ajouter des gens à un projet en retard le retarde davantage. ↩

10. Hackman, J. R. (2002). *Leading Teams : Setting the Stage for Great Performances*. Boston : Harvard Business School Press https://openlibrary.org/books/OL8754168M/Leading_Teams ; Wheelan, S. A. (2009). « Group Size, Group Development, and Group Productivity ». *Small Group Research*, 40(2), 247-262 https://doi.org/10.1177/1046496408328703 ; Allen, N. J. & Hecht, T. D. (2004). « The "romance of teams" ». *Journal of Occupational and Organizational Psychology*, 77(4), 439-461 https://doi.org/10.1348/0963179042596469 . Sur la paresse sociale : Latané, B., Williams, K. & Harkins, S. (1979). « Many Hands Make Light the Work ». *Journal of Personality and Social Psychology*, 37(6), 822-832 https://doi.org/10.1037/0022-3514.37.6.822 . ↩

11. Conway, M. E. (1968). « How Do Committees Invent ? ». *Datamation*, 14(5), 28-31 https://www.melconway.com/Home/Committees_Paper.html : « les organisations qui conçoivent des systèmes sont contraintes de produire des conceptions qui copient leurs structures de communication. » Le titre de ce chapitre lui fait écho. ↩

12. Parnas, D. L. (1972), *op. cit.* https://doi.org/10.1145/361598.361623 — la correspondance entre la structure des modules et celle de l'organisation qui les produit est esquissée dans l'article original. Pour la pratique contemporaine : Skelton, M. & Pais, M. (2019). *Team Topologies : Organizing Business and Technology Teams for Fast Flow*. Portland : IT Revolution Press https://books.google.com/books/about/Team_Topologies.ht

ml?id=Pj-IDwAAQBAJ — types d'équipes, modes d'interaction, et la « manœuvre de Conway inverse » (concevoir l'organisation pour induire l'architecture visée). ↩

13. Comptages effectués le 22 juillet 2026 sur le dépôt https://github.com/Darktable-org/Darktable via l'API GitHub. Fils nouveaux : 2 256, 1 946 puis 2 400 *issues* et *pull requests* créées sur les fenêtres juillet-juin 2023-2024, 2024-2025 et 2025-2026. Messages : 473 commentaires du 1er au 22 juillet 2026 (21,5 par jour, dont 2,6 émis par le robot d'intégration continue) ; relectures ligne à ligne : 3,4 par jour sur la même fenêtre ; clôtures et fusions : 2 196 sur juillet 2025-juin 2026. Contributeurs : historique git complet, commits hors fusions, auteurs dédupliqués par nom, du 1er janvier 2019 au 22 juillet 2026 — 2026 est une année partielle de 6 mois et demi. ↩

14. Journal du salon Matrix public du projet (pont du canal IRC), export du 23 juillet 2026, échange du 18 janvier 2023 entre 12 h 15 et 12 h 20. Le fondateur y poste deux actions : « *doesn't follow Darktable development so closely* » et « *goes read the github issue* » ; un contributeur enchaîne « if you don't read the issues, then there are no issues », puis plaisante sur le fait de ne pas se connecter au système de tickets de son propre employeur. Registre : c'est de l'auto-dérision, non un rapport factuel — je la cite comme aphorisme qui nomme un mécanisme, non comme témoignage. ↩

15. Même journal, 25 mars 2022. Le fondateur : « reading backlog is bad for your health » ; et, le même jour, ma propre

remarque : « I have spent the past 4 years trying to convince ppl that IRC is incredibly unfair to those who don't [have backlog] ». Le motif « je n'ai pas lu l'historique » est omniprésent chez les habitués du salon la même année. ↩

16. Journal du salon Matrix public du projet (export du 23 juillet 2026, voir chapitre « Le cercle fermé ») : « nobody looks at gh anyway », 2 juillet 2022 ; échange du 1er décembre 2024 — l'empaqueteur Debian : « I would prefer that non-bugs are closed with an explanation and that open bugs stay open… bugs accumulating is the lesser evil » ; le mainteneur du robot : « Would you prefer that such issues never get closed due to age and their number grows ? » (notre traduction dans le corps du texte). ↩

17. Même date d'arrêt, mêmes sources. Échantillon de traitement : les 1 000 derniers fils (19 mars-22 juillet 2026 ; 345 *issues*, 655 propositions de code). Latences : médiane de clôture 1,3 jour sur les *issues* fermées (53 %), 90e centile 22 jours ; fusion des propositions en 0,4 jour médian. Étiquetage : 44 % sans étiquette ; « no-issue-activity » (le robot de péremption) = 55 occurrences, première étiquette du registre ; « reproduce: confirmed » = 19. Encours au 22 juillet 2026 : 535 *issues* et 84 propositions ouvertes ; solde net sur 12 mois : +167 *issues*. Les séries d'*issues* commencent en 2019 parce que le projet a migré son suivi de bugs d'un Redmine auto-hébergé vers GitHub fin 2018 (l'ancien traqueur redirige depuis décembre 2019) — la borne est celle de l'archive, pas un choix. Battement : répartition hebdomadaire des créations de fils plate (9,6 à 17,7 % par

jour) ; commits 2024-2026 (n = 5 652, hors fusions) : 31,5 % le week-end, 40,1 % entre 18 h et 2 h, heure locale de l'auteur telle qu'enregistrée par git. Péremption : 2 589 *issues* portent l'étiquette du robot d'inactivité, sur 9 102 créées de 2018 au 22 juillet 2026 (28 %) ; fermetures étiquetées par année, de 2020 à 2026 : 424, 476, 132, 574, 297, 222, 153 — soit 27 à 40 % des fermetures de chaque année (premières fermetures étiquetées observées en 2020). Carte de chaleur : commits hors fusions 2016-2026 (n = 26 865), heure locale de l'auteur. Les séries d'effectifs, de départs et le pouls mensuel sont au chapitre « Livrer, pas larguer ». ↩

18. Freeman, J. (1972). « The Tyranny of Structurelessness ». *Berkeley Journal of Sociology*, 17, 151-164 https://www.jofreeman.com/joreen/tyranny.htm — écrit depuis l'expérience du mouvement de libération des femmes ; le texte circule depuis sous forme de brochure dans les mouvements militants, où la leçon se réapprend à chaque génération. ↩

19. O'Mahony, S. & Ferraro, F. (2007). « The Emergence of Governance in an Open Source Community ». *Academy of Management Journal*, 50(5), 1079-1106 https://doi.org/10.5465/amj.2007.27169153 — étude longitudinale de Debian : la légitimité s'y construit et s'y concède dans le temps, et la communauté a bâti une autorité formelle puis l'a bornée par des mécanismes démocratiques. ↩

20. Janis, I. L. (1972). *Victims of Groupthink*. Boston : Houghton Mifflin.

https://openlibrary.org/books/OL19773076M/Victims_of_groupthink Le mécanisme exige la cohésion et la recherche de l'unanimité — un comité sans cohésion produit du bruit, pas de la pensée de groupe ; les deux pathologies encadrent la même absence de méthode. ↩

21. Edmondson, A. (1999). « Psychological Safety and Learning Behavior in Work Teams ». *Administrative Science Quarterly*, 44(2), 350-383 https://doi.org/10.2307/2666999 . ↩

22. Raymond, E. S. (1999). *The Cathedral and the Bazaar.* Sebastopol : O'Reilly (essai initial 1997) https://www.catb.org/~esr/writings/cathedral-bazaar/ . Leçon 1 : « tout bon logiciel commence par gratter la démangeaison personnelle d'un développeur » ; leçon 7 : « livrez tôt, livrez souvent » ; leçon 8 : « avec suffisamment d'yeux, tous les bugs sont superficiels » (loi dite de Linus). ↩

23. Schweik, C. M. & English, R. C. (2012). *Internet Success : A Study of Open-Source Software Commons.* Cambridge, MA : MIT Press https://direct.mit.edu/books/monograph/3364/Internet-SuccessA-Study-of-Open-Source-Software — 174 333 projets SourceForge et 1 403 développeurs interrogés : les prédicteurs du succès sont le leadership par l'exemple, la vision claire et les objectifs clairement énoncés ; la gouvernance des projets qui réussissent est légère et informelle au début, les institutions formelles ne comptant que plus tard. ↩

21. Le travailleur invisible

Tout au long de ce livre, on a pensé la conception au service de l'utilisateur. Ce chapitre retourne l'objectif, et il énonce la thèse qui m'a demandé le plus d'années : **la qualité de vie de celui qui fait le travail est un critère de conception aussi.** C'est le grand angle mort. On mesure tout — les clics, les temps, les distributions — et l'on ignore à quel point il peut être pénible de simplement accomplir le travail. J'ai mis longtemps à le voir, par biais de technicien : ma formation d'ingénieur accordait un poids démesuré à l'aspect technique des choses. Or le milieu de travail, la façon de travailler et le confort de travail comptent au moins autant que l'intelligence déployée à trouver de bonnes solutions — parce que derrière chaque problème technique, il y a un humain, y compris celui qui écrit le logiciel. La formule ouvre et ferme ce livre, ce chapitre la prend au sérieux.

Il faut le poser tout de suite, car la suite y joue son équilibre : la discipline décrite dans ces pages est une discipline du soin dû à autrui, et un soin qui épuise celui qui le donne n'est pas une vertu mal récompensée — c'est une exploitation de plus, avec une meilleure conscience. Du reste, les conditions de travail sont aussi un facteur d'attraction ou de répulsion de la main-d'œuvre, et soigner le travailleur permet d'éviter de voir ses effectifs fondre avec le temps, comme nous l'avons vu aux chapitres précédents avec Darktable.

Pour rendre le travailleur visible, il faut d'abord rendre le travail visible, car le public n'en voit que la pointe : les commits, l'agitation du dépôt. Voici l'inventaire réel de mon propre cas :

- **le développement**, et la relecture du code des autres ;
- **la documentation** : des dizaines de pages, entretenues en deux langues ;
- **l'enseignement** : des dizaines d'heures de tutoriels vidéo, pour montrer quel module sert à quoi, dans quel contexte, avec quel bénéfice ;
- **le support** : les forums, et tous les canaux où l'on vous trouve ;
- **l'intendance** : le tri des rapports d'anomalie, l'empaquetage des versions, l'infrastructure de compilation, la modération ;
- **et le flux de communication qui enveloppe le tout** : pour la seule année 2021, GitHub m'a envoyé plus de 4 000 courriels ; sur les 8 premiers mois de 2022, le forum de la communauté en a ajouté près de 700 ; et l'on me sollicitait en parallèle sur YouTube, Reddit, Matrix, Telegram et par courriel direct.

Hors des projets adossés à des fondations, ce niveau de support et de documentation n'existe tout simplement pas dans le logiciel libre — et là où il existe, ce ne sont généralement pas les développeurs qui le portent[1]. Je portais les deux, devenu le visage public du projet sans l'avoir décidé, et sans personnel pour faire écran aux sollicitations. L'économie de l'ensemble tient en deux chiffres : je n'ai jamais compté plus de 240 donateurs, quand mes deux enquêtes réunissaient plus de 2 000 répondants, dont la dépense photographique médiane déclarée avoisinait 500 euros

par an. Les gens ne respectent le travail que lorsqu'il leur a été facturé au prix fort.

Le premier critère de confort est un régime de travail, et le chapitre « Finir et maintenir » en a posé la mécanique : le travail doit capitaliser. Le régime à fuir à tout prix est celui où plus on travaille, plus il faut travailler — où corriger un défaut en crée un autre, où tout s'achète par de la dette technique, où l'on ne thésaurise pas. Bien travailler, c'est accumuler : des outils, des méthodes, de la compréhension. Chaque heure passée à se doter d'un outil de diagnostic, d'une collecte propre des rapports d'anomalie, d'un test qui verrouille un acquis, est une heure qui se rembourse ensuite indéfiniment ; on bouche les trous à mesure qu'on avance, et l'édifice devient étanche. Plus on travaille, moins on *a besoin* de travailler. Et la meilleure façon d'alléger le travail n'est pas toujours d'écrire du code : une page de documentation qui tarit une source de questions, une courte vidéo, un menu réorganisé font le même office. La règle de raisonnement est simple : penser du point de vue de celui qui fera tourner la machine dans 6 mois — le plus souvent soi-même. Sans méthode, sans outils, sans objectif commun ni liste de tâches partagée, une équipe de 10 ou 15 personnes se retrouve sur des sables mouvants, à encaisser un imprévu après l'autre : toujours dans la réaction, jamais dans l'action, encore moins dans la réflexion. À l'inverse, les situations où le travail engendre plus de travail sont les révélateurs de machines infernales en plein emballement, et c'est sur la méthode de travail qu'il faut agir avant qu'il ne soit trop tard.

Le deuxième critère est le contrôle de l'exposition, et il faut d'abord décrire l'asymétrie, car elle est invisible de l'extérieur. Dans une entreprise, le développement se fait à l'abri : quand on veut du retour utilisateur, on organise un panel — quelques personnes devant l'outil, derrière une vitre, à un moment choisi, avec des gens choisis. On garde la main sur la temporalité. Dans le logiciel libre, tout est public, tout le temps ; on est exposé en permanence, et jamais au moment qu'on aurait décidé. Et cette perte de la temporalité choisie joue dans les deux sens : on ne choisit pas quand le retour du monde arrive — on verra plus bas ce qu'il pèse — mais on ne choisit pas davantage quand cesser de veiller, car tout se décide en public, à toute heure, et quitter le flux, c'est laisser le projet se décider sans soi. Deux scènes vécues donnent la texture de cette servitude de présence. Je passe un mois dans ma grotte à développer un espace de couleur perceptuel, un projet de recherche long et exigeant ; j'en ressors pour découvrir qu'un nouveau système labyrinthique de filtrage des images a été fusionné en mon absence, et pour m'entendre dire que j'arrive trop tard pour m'y opposer. Je prends 3 semaines de vacances en août ; à mon retour, ma relecture d'un changement mathématique sur ce même espace de couleur a été contournée — il fallait avancer vite. Pour quelle urgence, exactement ? Dans ce régime, s'absenter n'est pas se reposer : c'est accumuler du retard sur des décisions déjà prises. S'y ajoute l'asymétrie des statuts : pour vos utilisateurs, c'est un loisir — le soir, le week-end ; pour vous, c'est le travail. Le retour arrive donc quand cela les arrange, au milieu de choses (plus) importantes, et il faut tout lire pour trier, puis expliquer, justifier, défendre chaque choix. Multipliez par une foule : vous affrontez, seul, plus de temps et d'énergie

cumulés que vous n'en posséderez jamais. Les dons financiers n'allègent pas cette charge, ils l'alourdissent d'attentes — et une étude longitudinale des plateformes de dons confirme leur impuissance structurelle : un effet significatif sur l'activité à court terme, aucun à long terme ; l'argent sans structure achète des attentes, pas de la soutenabilité[2]. La recherche a donné un nom à cette économie : l'attention du mainteneur est une ressource commune surexploitée — chacun y puise comme si elle était gratuite et infinie, et personne n'est comptable de son épuisement[3]. Elle a aussi documenté le mécanisme qui transforme cette exposition en épuisement : les interactions toxiques — l'agressivité, le mépris, l'exigence sans gratitude — sont identifiées comme un moteur direct du burnout des mainteneurs[4]. Le mal est d'époque : les créateurs étaient autrefois protégés par des intermédiaires — un agent, un éditeur — qui filtraient le courrier et absorbaient les coups ; aujourd'hui on publie son travail directement, une boîte à commentaires accrochée dessous, et l'on est à découvert par défaut. La communauté est de surcroît presque exclusivement masculine, et l'échange y prend souvent la forme d'un rapport de force : il faut se défendre en permanence — le chapitre « Ce que l'entre-soi produit » dira qui compose cette communauté et pourquoi cela compte.

Dans ce flux, une catégorie mérite son paragraphe : les idées. Les utilisateurs en ont à revendre, mais manquent des compétences pour en faire des prototypes. Il faut comprendre une chose à leur sujet : les idées ne valent rien ; ce qui coûte, c'est de les réaliser. Neuf de vos propres idées sur 10 sont mauvaises, et vous ne

l'apprenez qu'en essayant de les construire ; celui qui n'a pas les moyens de réaliser ses idées ne reçoit jamais ce rappel à la réalité. Persuadé qu'avoir eu une idée fait de lui quelqu'un d'exceptionnel, il vous tend l'idée *et* la charge de réaliser son rêve. Une idée ne coûte rien ; livrer un produit fini, utilisable en production — pas un prototype à moitié fonctionnel —, coûte tout. Le tri des retours du chapitre « Écouter sans obéir » prend ici sa fonction protectrice : le témoignage d'usage se traite, le fantasme délégué se décline poliment — et les chiffres du chapitre « Compter les absents » servent aussi de bouclier, car renvoyer une exigence bruyante à la réalité de ses 5 % est une manière de s'acheter la paix, de se protéger.

Vient alors la seule règle que je tienne pour vitale, apprise de l'erreur que je ne souhaite à personne : ne soyez pas trop disponible — c'est la porte ouverte à l'épuisement. La disponibilité est une ressource finie ; il faut la traiter comme telle. C'est vous qui décidez quand vous lisez les retours, pas ceux qui les envoient : regroupez-les, filtrez-les, ne répondez pas dans l'instant, ne traitez aucun message comme une urgence tant qu'il n'en est pas une. Temporiser le traitement des problèmes n'est pas de la paresse, c'est même tout le contraire. Lorsqu'on change un élément critique du code, on est à peu près sûr de casser quelque chose, la question est : quoi ? Un rapport de bug unique est rarement suffisant pour remonter à la source de problème. Laisser le bug ouvert un moment donne le temps à d'autres rapports de bug d'arriver, qui vont trouver des problèmes se manifestant à d'autres endroits mais pointant vers la même origine. On obtient alors un cluster de bugs qui voient le même

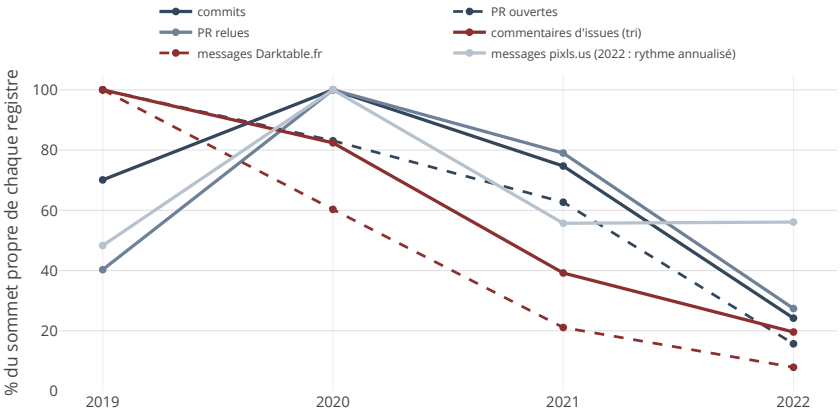
problème depuis des points de vue différents et, par une sorte de triangulation, permettent souvent de remonter à l'origine bien plus facilement. À condition de les lire tous…

L'autre argument pour la temporisation est que le travail de conception a aussi son rythme propre, que la tradition des ateliers appelle l'horaire du faiseur : les tâches créatives et analytiques n'existent que par blocs de plusieurs heures, et une seule interruption mal placée détruit le bloc entier[5] — l'urgence permanente n'est donc pas seulement pénible, elle est incompatible avec la nature du travail. On ne conçoit rien de bon en état d'urgence permanente ; se ménager de la marge n'est pas un luxe, c'est une condition de la conception.

La protection peut aller jusqu'à s'outiller — le principe ici, le récit complet au chapitre suivant : tous les problèmes ne se résolvent pas par du code, mais certains problèmes de communication, si. En 2022, le volume de sollicitations avait atteint le point de non-retour : la majorité des messages humains me réduisaient à un moteur de recherche en chair et en os, sommé de retrouver dans ma propre documentation des contenus que j'avais déjà écrits, parfois dans les deux langues. J'ai fini par construire la machine qui manquait — un secrétaire de tri, puis un moteur de recherche sur mes propres contenus ; le chapitre « Travailler pour le travail » raconte comment. Seule la doctrine importe ici, et c'est celle du chapitre « Finir et maintenir » étendue à la communication : une heure passée à outiller le filtrage se rembourse indéfiniment ; une heure passée à répondre une énième fois à une question documentée est perdue pour tout le

monde — y compris pour celui qui la posait, car la réponse fatiguée est une mauvaise réponse.

Il faut aussi donner ses unités au coût, car « épuisement » reste abstrait tant qu'on ne le chiffre pas dans un corps et un calendrier. Mon premier syndrome du canal carpien date d'il y a plus de 10 ans : taper au clavier et manipuler la souris est un travail physique, et la lésion de travail répétitif du chapitre « Poser le problème » n'épargne pas celui qui fabrique l'outil. La dépression induite par le logiciel figure, nommée ainsi, dans mes journaux de chantier. Et le grand nettoyage de la photothèque (table lumineuse) — près de 300 commits sur 4 ans pour défaire des contournements dont la totalité datait d'après 2020 — m'a coûté 4 ans de ma vie, sans aucun plaisir. Et l'épuisement lui-même se lit dans les registres de contribution, avec une précision qui m'a surpris moi-même quand je l'ai mesuré. L'ordre des abandons est la signature : en 2021, mon code tient — 210 commits, une année presque normale — mais mes contributions d'interaction sont déjà réduites de moitié : le tri des *issues*, les forums, tout ce qui expose. On rogne d'abord la surface de contact pour protéger le geste. En 2022, le geste cède à son tour — commits, propositions, relectures : en recul de deux tiers partout — avec une dernière flambée en mai, dépensée dans le conflit des filtres, pendant que mes messages sur le salon explosaient. Puis la falaise : un commit en juillet, dernier passage sur le forum francophone le 24, silence en septembre[6]. Je suis, mesurablement, l'un des points de 2022 de mon propre graphique des ciseaux — un profil à plein régime puis plus rien, pas un déclin graduel.



L'épuisement, mesuré dans les registres de contribution, 2019-2022 — chaque registre en pourcentage de son propre sommet. En bleu et gris, le geste : commits, propositions de code, relectures. En rouge, la surface d'exposition : le tri des issues et le forum francophone. L'ordre des abandons est la signature : les registres d'exposition sont coupés de moitié dès 2021, un an avant que le code ne cède — on rogne d'abord ce qui expose pour protéger ce qui produit. Le forum anglophone (gris clair, 2022 en rythme annualisé sur le premier semestre) tient jusqu'à la sortie : c'est le canal d'enseignement, abandonné en dernier. Mesures de contribution, pas diagnostic clinique ; sources en note.

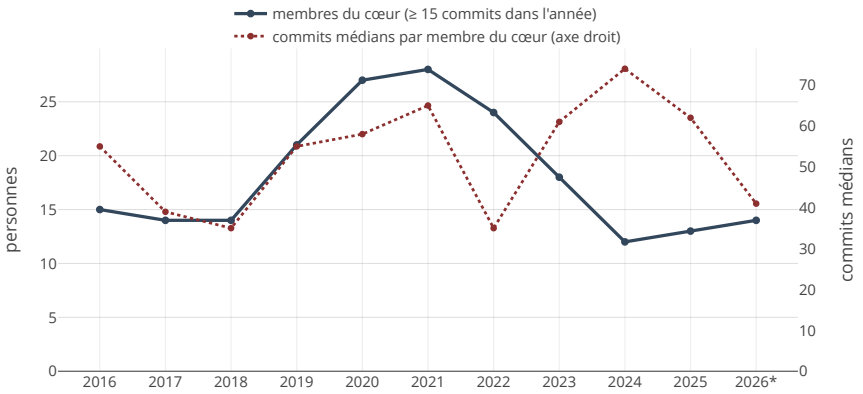
Il y a une différence énorme entre manquer de temps pour bien faire et consommer des heures de travail humain à aggraver les choses ; ceux qui introduisent les régressions que d'autres réparent péniblement portent la responsabilité des conséquences. C'est de la comptabilité : les heures des réparateurs sont le budget du chapitre « Finir et maintenir », et quelqu'un les dépense.

Le registre du projet permet de suivre ces sorties à l'échelle, et leur *forme* dit ce que les moyennes cachent. J'ai reconstitué le profil de fin de course des 22 membres du cœur — 15 commits ou plus dans leur dernière année d'activité — qui ont quitté Darktable depuis 2016. Trois seulement s'éteignent en pente douce, l'affaiblissement graduel qu'on prête d'ordinaire à l'épuisement. Tous les autres partent d'un coup, et il faut regarder quelques-uns de ces électrocardiogrammes pour comprendre ce que « d'un coup » veut dire :

- **un pilier de la première décennie** — 11 ans de présence, plus de 2 000 commits : ses 3 derniers trimestres pleins *accélèrent*, 10, 18, 27 commits, puis le registre s'arrête net — deux commits résiduels, et plus jamais rien ;
- **un météore** — 3 ans, 800 commits, un régime de 150 à 170 commits par trimestre : encore 157 dans son avant-dernier trimestre, 9 dans le dernier, puis le silence ;
- **un membre de 5 ans** — une dernière pointe à une centaine de commits au premier trimestre 2022, deux trimestres résiduels de 15 puis 3 commits, et la sortie à l'été 2022 ;
- **le doyen** — 14 ans de présence, près de 1 400 commits : ses deux dernières années ne sont plus qu'un clignotement — un trimestre à 19 commits fin 2023, un an de quasi-silence, un dernier battement à 19 début 2025, puis rien.

À quoi s'ajoutent une dizaine de flambées : presque rien, un ou deux trimestres intenses — 20 commits d'un coup pour quelqu'un qu'on n'avait jamais vu —, puis la disparition[7]. La leçon n'est pas psychologique, elle est structurelle : il n'existe aucun poste à mi-temps dans un projet pareil. On est du cœur ou l'on n'est rien ;

faute de pouvoir ralentir, on casse. Et pendant ce temps, ceux qui restent absorbent la charge de ceux qui sont partis : le cœur a fondu de moitié entre 2021 et 2024, et les commits médians par survivant ont plus que doublé sur la même période — 35 en 2022, 74 en 2024, le plus haut niveau de la décennie[7]. Moins de bras, plus de charge par bras, et aucun régime intermédiaire entre le plein feu et la sortie : ce système ne fabrique pas des gens fatigués — il fabrique des ruptures. Et la qualité du code produit s'en ressent forcément : la quantité de code qui double par développeur a forcément un prix.



Les ciseaux : effectif du cœur de Darktable (≥ 15 commits dans l'année) contre commits médians par membre du cœur. Quand l'effectif fond de moitié (2021-2024), la charge médiane du survivant double — le creux de 2022 est l'année du conflit, où même la production des présents s'effondre. Commits hors fusions, auteurs dédupliqués ; 2026 est partiel.

Et il faut compter ce que ces ruptures emportent, car cela se lit dans les fichiers : 39 % des fichiers sources du projet ont aujourd'hui perdu leur auteur principal — celui qui en a écrit l'essentiel ne commet plus — et 6 personnes détiennent à elles seules la paternité majoritaire de la moitié de la base de code[8]. Un arrêt net n'emporte pas qu'un travailleur : il emporte la connaissance des fichiers qu'il était seul à porter — sans passation, puisque partir d'un coup, c'est précisément partir sans passation et que le projet ne laisse pas de documentation développeur (qui, du reste, augmenterait encore la charge de travail).

Il faut enfin dire ce qu'est le burnout, et ce qu'il n'est pas. Ce n'est pas une faiblesse individuelle : c'est la sortie normale du système décrit dans ce chapitre — exposition permanente, asymétrie des statuts, extraction d'attention sans compte, absence de filtres — appliqué à un humain ordinaire pendant assez longtemps. Les chiffres du chapitre « Finir et maintenir » en donnent la prévalence ; ce chapitre en donne le mécanisme. Et depuis 2024, l'argument a changé de catégorie. La porte dérobée découverte dans xz-utils — une bibliothèque de compression présente dans à peu près tous les systèmes Linux du monde — n'a pas été insérée par un exploit technique : elle l'a été par ingénierie sociale, sur un mainteneur solitaire et épuisé, harcelé pendant des années par de faux comptes qui le pressaient de céder la maintenance à son futur attaquant[9]. L'épuisement du travailleur invisible n'est plus seulement une question de bien-être : c'est une surface d'attaque de la chaîne d'approvisionnement logicielle mondiale. Le confort du mainteneur est une exigence de sécurité.

L'objection n'a rien d'hypothétique : elle m'a été opposée publiquement, pendant des années, et notamment par un modérateur du forum de la communauté — « personne ne le force à rester » quand je décrivais les conditions ; et, à l'annonce de mon retrait pour épuisement, la victime fut renvoyée à son privilège supposé : « il parle d'avoir eu trop de temps libre — mais qui a eu presque six ans pour, l'essentiel du temps, écrire du code ? »[10]. La forme chimiquement pure de l'argument a été donnée ailleurs, en réponse à quelqu'un qui rappelait qu'on ne pousse pas les gens au burnout sans conséquences : « Qui l'a poussé ? Il s'est poussé lui-même. Tout était bénévole, tout était des choses qu'il voulait faire. Personne ne l'a supplié, personne ne lui a tenu un pistolet sur la tempe. »[11] Le bénévolat est un choix ; personne ne vous doit du confort ; si c'est trop dur, arrêtez. Trois réponses. D'abord, le travail gratuit reste du travail : Terranova l'a décrit il y a 25 ans — un travail « donné volontairement et non payé, joui et exploité », dont l'économie numérique entière se nourrit[12] ; le choix d'entrer ne rend pas l'extraction équitable, et l'industrie qui embarque ce code dans ses produits n'est pas un tiers innocent au différend. Ensuite, l'argument prouve trop : s'il suffisait que le travailleur soit volontaire pour que ses conditions soient à l'abri de toute critique, aucune organisation bénévole ne pourrait jamais être tenue pour maltraitante : les conditions sont une propriété du système, pas du consentement. Enfin, même celui que le sort du bénévole indifférence a un intérêt d'assurance : xz-utils tournait chez lui aussi. Le confort du travailleur n'est pas un supplément d'âme que les projets s'offriront quand tout ira bien ; c'est une condition de survie du logiciel, au même titre que ses tests.

Mais le fait est là, et les registres publics le caractérisent : la « communauté » n'est pas un groupe de personnes qui prennent soin les unes des autres — ce que la sociologie appelle une communauté depuis Tönnies (1887) : des liens entretenus pour eux-mêmes, où la personne compte plus que sa fonction[13] —, mais un groupe d'intérêt qui s'enthousiasme des mêmes choses, veut son logiciel-jouet bardé de nouveautés excitantes à dates fixes, et se moque des conditions de travail de ceux qui le produisent, tant que la façade tient. Le bénévolat y est instrumentalisé tour à tour :

- pour excuser l'incompétence et la négligence du travailleur — c'est du bénévolat, on fait ce qu'on peut ;
- pour évacuer toute responsabilité du travailleur face aux conséquences de son travail, sur les utilisateurs comme sur ses collègues ;
- pour évacuer toute responsabilité de la communauté dans les conditions de travail du travailleur ;
- comme un consentement global et sans réserve à toutes les conditions de travail, tant que le travailleur reste. Sinon il est prié de partir, mais en laissant le produit de son travail pour la jouissance de tous, dans une forme d'aliénation marxiste que Marx n'avait pas prévue[14].

Les plus crus de ces usages viennent d'être cités, verbatim et archivés ; l'inventaire complet en compte 22 occurrences, sur deux plateformes, en quatre ans[10]. C'est le double discours endémique du logiciel libre, celui que le prologue annonçait : façade libertaire, bienveillante, émancipatrice, collectiviste — jusqu'à ce qu'on le critique, qu'on le mette face à ses

contradictions, ou qu'on se fatigue d'éponger sans fin, faute de structure et de méthode, l'incompétence de « collègues » qui ne s'améliorent pas. Alors il montre sa vraie nature : individualiste, consumériste, exploiteur et violent. Et il se maintient dans la croyance défaitiste que c'est comme ça partout et qu'on n'y peut rien — alors que la dégradation d'un projet comme Darktable est mesurable, datée, imputable :

1. à des décisions prises à un instant donné et jamais réévaluées depuis qu'on peut en mesurer les effets — le chapitre « Livrer, pas larguer » a daté la décision, et le chapitre « L'ambition non héroïque » a montré le diagnostic des fondateurs eux-mêmes, resté sans suite ;
2. à des individus qui refusent de mesurer l'impact de leurs contributions sur la dette technique et la stabilité de l'ensemble — le chapitre « Le culte du cargo » en a fait l'arithmétique — et qui se retranchent derrière la croyance qu'il n'était pas possible de faire mieux : tautologie s'il en est ;
3. à des croyances fausses sur ce qui entretient la motivation des contributeurs — ne rien refuser, ne rien planifier, livrer deux versions par an au lieu d'une — quand les projets qui durent font l'inverse : le taux de refus normal d'un projet sain est mesuré au chapitre « L'ambition non héroïque », et ce qui sépare les projets libres qui vivent de ceux qui meurent, mesuré sur plus de cent mille projets, tient à la gouvernance et aux objectifs énoncés — pas à la nature de la tâche[15].

Rien de tout cela n'est une propriété de la tâche à accomplir ; tout est propriété des croyances des personnes en présence, et de leur façon de travailler. Rien de tout cela, donc, n'est une fatalité.

Ce chapitre a fait du confort du travailleur un critère ; reste à montrer qu'on peut le tenir, concrètement, dans un projet réel. Tout ce qui précède — l'exposition, la disponibilité finie, le burnout comme propriété du système — appelle une réponse pratique : une organisation qui rende le travail fiable et, à terme, moins lourd. C'est l'objet du chapitre suivant : travailler pour ne plus avoir à retravailler.

---

1. Mesuré sur le salon Matrix public du projet (journal cité au chapitre « Le cercle fermé »), en comptant les réponses adressées — citations et interpellations nominales. En 2022, sur 4 905 réponses adressées, les 3 premiers répondants sont un utilisateur expérimenté (30 %), moi (14,5 %, 85 interlocuteurs distincts — en sus du développement), et le modérateur du forum (11 %) ; le fondateur du projet : 2,5 % ; le fusionneur principal : zéro réponse adressée, 36 messages en 4 ans et demi. Les années suivantes, le support est porté par des utilisateurs aguerris et des modérateurs — jamais par ceux qui décident de la conception. Heuristique : les réponses non adressées échappent au compte ; le classement relatif, lui, est robuste. L'agrégat des deux lieux publics donne l'arc complet de cette charge : sur le forum de la communauté, mes messages passent de 44 (2017) à 191, 401, puis 830 au sommet de 2020, 462 en 2021 et 233 sur ma dernière demi-année — 2 161 en tout (comptage intégral par

l'API, toutes catégories : entraide, annonces et débats mêlés) ; sur le salon, 709 réponses adressées sur les 7 mois documentés de 2022. Et il faut borner la mesure dans l'autre sens, car elle se vérifie : l'entraide *en largeur* — être présent dans le plus de fils possible — a toujours été portée par d'autres, le modérateur du forum en tête chaque année de 2017 à 2025 ; sur ce classement-là, je n'apparais dans les 5 premiers qu'une seule année (2020). Ma forme à moi était la profondeur et l'enseignement : 37 fils ouverts, dont les fils de référence — la FAQ filmic et ses 222 messages, le fil du *tone equalizer* — que la mesure par fils ne compte pas, et que le salon citait encore en réponse des années après mon départ. Le forum francophone (Darktable.fr) documente la période antérieure : 1 143 messages de 2016 à 2022, dont 493 pour la seule année 2018 — 4e répondant du forum cette année-là, 6e en 2019 — avant que mon activité ne migre vers les canaux anglophones ; dernier passage enregistré par la plateforme : le 24 juillet 2022. Sur le domaine expert — la science de la couleur, filtrée par mots-clés —, un tiers des messages du salon étaient de moi, 3 fois plus que le contributeur suivant ; et des années après mon départ, la réponse type du salon aux questions de couleur restait un lien vers mes vidéos. Après septembre 2022 : zéro partout — et l'entraide totale du salon, toutes personnes confondues, tombe des deux tiers l'année suivante, sans jamais remonter (chapitre « Le cercle fermé »). ↩

2. Zhang *et al.* (2025). Étude longitudinale des dons aux mainteneurs (GitHub Sponsors). *Journal of Software : Evolution and Process* https://doi.org/10.1002/smr.70033 —

effet significatif sur l'activité à court terme, aucun effet à long terme ; la littérature sur les parrainages note aussi le risque d'éviction de la motivation intrinsèque. ↩

3. Eghbal, N. (2020). *Working in Public : The Making and Maintenance of Open Source Software*. San Francisco : Stripe Press https://press.stripe.com/working-in-public — l'attention du mainteneur comme ressource commune surexploitée est le cœur de son analyse. Voir aussi, de la même autrice, *Roads and Bridges : The Unseen Labor Behind Our Digital Infrastructure*. Ford Foundation, 2016. https://www.fordfoundation.org/learning/library/research-reports/roads-and-bridges-the-unseen-labor-behind-our-digital-infrastructure/ ↩

4. Raman, N., Cao, M., Tsvetkov, Y., Kästner, C. & Vasilescu, B. (2020). « Stress and Burnout in Open Source : Toward Finding, Understanding, and Mitigating Unhealthy Interactions ». *ICSE-NIER 2020*, ACM https://doi.org/10.1145/3377816.3381732 — le mécanisme interactions toxiques ⟶ burnout ; l'étude ne fournit pas de chiffres de prévalence, et n'est pas citée ici pour cela. ↩

5. Graham, P. (2009). « Maker's Schedule, Manager's Schedule ». Essai en ligne (paulgraham.com) https://paulgraham.com/makersschedule.html — l'horaire du faiseur par demi-journées, contre l'horaire du manager par heures : une réunion mal placée coûte une demi-journée de fabrication, pas une heure. ↩

6. Mes contributions, 2019-2020-2021-2022, sources : historique git (alias dédupliqués), API GitHub, API des deux forums, journal du salon. Commits : 197, 281, 210, 68 ; dont correctifs : 68, 39, 38, 20 ; PR ouvertes : 83, 69, 52, 13 ; PR relues : 25, 62, 49, 17 ; commentaires d'issues : 199, 164, 78, 39 — le tri s'effondre dès 2021 (− 52 %), un an avant le code ; posts Darktable.fr : 317, 191, 67, 25 ; posts pixls : 401, 830, 462, puis 233 sur le premier semestre 2022. Profil mensuel 2022 des commits : 18, 4, 0, 8, 32 (mai, le mois du conflit des filtres), 5, 1, puis rien. Autodéclarations datées, confrontées aux mesures : « deux semaines de vacances depuis Noël » (salon, 8 août 2022) ; « je n'ai le temps de rien relire » (même soir — 17 relectures cette année-là, contre 62 en 2020). Mesure de contribution, pas diagnostic clinique. ↩

7. Historique git du dépôt Darktable, commits hors fusions, auteurs dédupliqués, arrêté au 22 juillet 2026. Départs du cœur : 15 commits ou plus l'année n−1, zéro commit l'année n, 2016-2025 (18 à 22 cas selon la méthode d'appariement des alias — les profils sont établis sur les 22). Classement sur les 8 derniers trimestres actifs, par le rapport entre les deux derniers et les deux premiers : 3 déclins graduels, 16 arrêts nets (dont environ 10 « flambées » : historique quasi nul, un ou deux trimestres intenses, disparition), 3 profils mixtes. Les 4 profils cités, 6 derniers trimestres (recalcul du 23 juillet 2026, stable selon l'appariement) : pilier 2010-2020, 2 127 commits, 4-10-18-27-0-2 ; météore 2014-2016, 841 commits, 109-110-49-172-157-9 ; membre 2018-2022, ≈ 550 commits, 33-22-27-101-15-3 ; doyen 2012-2025, ≈ 1 370 commits, 19-3-

0-2-0-19. La flambée citée : 20 commits, tous dans un unique trimestre de 2023, sans aucune activité avant ni après. Charge des survivants : commits médians des membres à 15 commits et plus — 55, 39, 35, 55, 58, 65, 35, 61, 74, 62 puis 41 pour 2016 ⟶ 2026 (partiel). Réserve de méthode : les commits ne mesurent ni la relecture ni le tri, et un profil d'arrêt net peut aussi relever d'un événement de vie ; c'est la rareté des déclins graduels, pas chaque cas individuel, qui porte la conclusion. ↩ ↩

8. Les 822 fichiers sources (C, C++, OpenCL) présents à la tête du dépôt Darktable, historique complet hors fusions, auteurs dédupliqués : 7 % n'ont qu'un seul auteur, 13 % au plus deux ; 323 (39 %) ont un auteur principal qui n'a plus rien commis depuis 2025 ; le plus petit groupe d'auteurs principaux couvrant la moitié des fichiers — le « facteur de bus » de la littérature — compte 6 personnes. Les décomptes pondèrent toute l'histoire du fichier : ils sous-estiment donc la perte récente. ↩

9. CVE-2024-3094 (mars 2024) : porte dérobée dans xz-utils/liblzma, insérée par un contributeur ayant obtenu la co-maintenance après une campagne de pression de plusieurs années — comptes multiples exhortant le mainteneur d'origine, qui se déclarait épuisé et en difficulté, à passer la main. Découverte par hasard avant déploiement général. https://nvd.nist.gov/vuln/detail/CVE-2024-3094 ↩

10. Verbatim, discuss.pixls.us. Fil « r/Darktable options and path forward », 2022-06-29 : « Yes, he had a lot of knowledge. But if he doesn't want to be here, nobody is forcing him to stay »

(https://discuss.pixls.us/t/r-Darktable-options-and-path-forward-and-user-point-of-view/31388/57 ; archivé : https://web.archive.org/web/20260721230849/https://discuss.pixls.us/t/r-Darktable-options-and-path-forward-and-user-point-of-view/31388/57) ; et fil « AP's new video on Ansel », 2026-06-20 : « He talks about having too much free time, but who had almost six years to mostly write code ? » (https://discuss.pixls.us/t/aps-new-video-on-ansel/58735/10 ; archivé : https://web.archive.org/web/20260721230927/https://discuss.pixls.us/t/aps-new-video-on-ansel/58735/10). Les deux messages émanent du même modérateur du forum. Inventaire complet, avec liens et contexte : 22 occurrences de cette rhétorique sur deux plateformes, 14 auteurs distincts, 2022-2026 (relevé du 22 juillet 2026). ↩ ↩

11. Verbatim, r/DarkTable (Reddit), fil « Ansel : Color controls finally correct », avril 2026 : « Who pushed him ? He pushed himself. It was all volunteer work, it was all things he wanted to do. Nobody begged him, nobody held a gun to his head. » (https://reddit.com/r/DarkTable/comments/1sdwjxo/comment/oeofv15/ ; archivé : https://web.archive.org/web/20260721231145/https://old.reddit.com/r/DarkTable/comments/1sdwjxo/ansel_color_controls_finally_correct/oeofv15/). Dans le même fil, un tiers releva la contradiction : « Quand il s'agit de travailler gratuitement, c'est son choix ; mais quand il s'agit de rendre service aux utilisateurs, il aurait soudain des responsabilités ? Il faut choisir. »

(https://reddit.com/r/DarkTable/comments/1sdwjxo/comment/oep7213/ ; archivé : https://web.archive.org/web/20260721231207/https://old.reddit.com/r/DarkTable/comments/1sdwjxo/ansel_color_controls_finally_correct/oep7213/). ↩

12. Terranova, T. (2000). « Free Labor : Producing Culture for the Digital Economy ». *Social Text*, 18(2), 33-58 https://doi.org/10.1215/01642472-18-2_63-33 — le travail libre, « donné volontairement et non payé, joui et exploité » (*voluntarily given and unwaged, enjoyed and exploited*). ↩

13. Tönnies, F. (1887). *Gemeinschaft und Gesellschaft* ; trad. fr. : *Communauté et société. Catégories fondamentales de la sociologie pure*, Paris : PUF, 2010. La communauté (*Gemeinschaft*) tient par des liens organiques entretenus pour eux-mêmes — parenté, voisinage, entraide — quand la société (*Gesellschaft*) agrège des intérêts individuels qui coexistent sans obligation mutuelle. Le vocabulaire du logiciel libre dit « communauté » ; ses registres décrivent une société. ↩

14. Marx, K. (1844). *Manuscrits économico-philosophiques de 1844*, premier manuscrit, « Le travail aliéné » (trad. fr. Paris : Éditions sociales) : le produit du travail devient pour le travailleur un objet étranger, une puissance indépendante de lui. Le bénévolat du logiciel libre y ajoute le cas que Marx n'avait pas envisagé : une aliénation sans salaire et sans contrainte, consentie — le travail « donné volontairement et non payé, joui et exploité » de Terranova (note ci-dessus),

poussé jusqu'à sa conclusion : le travailleur peut partir, le produit reste. ↩

15. Schweik, C. M. & English, R. C. (2012). *Internet Success : A Study of Open-Source Software Commons*. Cambridge, MA : MIT Press https://mitpress.mit.edu/9780262017251/internet-success/ — analyse de plus de 100 000 projets hébergés sur SourceForge ; les facteurs qui séparent les projets qui aboutissent de ceux qui s'abandonnent relèvent de la conduite du projet — leadership, vision, objectifs énoncés, utilité constatée — et non du domaine ou de la difficulté de la tâche. Déjà cité au chapitre « Le comité n'inventera rien ». ↩

22. Travailler pour le travail

Le chapitre précédent a posé un principe : le travail doit capitaliser, faute de quoi il dévore le travailleur. Capitaliser signifie ici que le travail doit produire des connaissances sur lui-même, des méthodes et des outils qui rendront le travail futur plus facile et plus rapide en utilisant l'expérience passée. Ce chapitre montre comment, et c'est le plus concret du livre : un compte rendu d'expérience plus qu'un argument.

Mais produire des connaissances sur son propre travail ne va pas de soi : cela suppose de travailler avec une méthode — quelque chose qui se valide, se falsifie et s'amende. Un travail que rien de tel n'encadre ne produit aucun savoir sur lui-même : chaque nouvelle tâche démarrée sans une idée préconcue de la séquence de sous-tâches à accomplir et à valider pour la mener à bien laisse le projet aussi ignorant qu'avant elle, car sans critère posé d'avance, impossible de dire ce qui a marché, ni pourquoi, ni ce que l'essai suivant devrait changer ou conserver. Et ce n'est pas une image, c'est un théorème : en optimisation, les résultats dits « no free lunch » établissent qu'en moyenne, sur l'ensemble des problèmes possibles, toutes les stratégies de recherche — tirage au hasard compris — ont exactement la même performance ; on ne fait mieux que le hasard qu'en incorporant à sa stratégie un savoir vrai sur la structure du problème[1]. La méthode n'est rien d'autre que ce savoir, accumulé, vérifié et transmissible — Simon décrivait déjà la conception comme une recherche dans un espace

de possibles, rendue praticable par les heuristiques qui l'élaguent[2] ; et la falsifiabilité qui fonde ce savoir a été posée dès le chapitre « Le mot et la chose ». Un collectif qui travaille sans méthode explore donc son espace de solutions comme un réseau de neurones en apprentissage : par essais aveugles, en masse. Simplement, il le fait à vitesse humaine. La machine peut se payer la force brute parce qu'elle essaie des millions de fois par seconde — le chapitre « L'intelligence en amont » a montré ce qu'on en tire quand on l'attelle à bon escient ; à l'échelle d'une équipe de bénévoles du soir, la même stratégie porte un autre nom : l'éternité. Dawkins en a fait la démonstration arithmétique restée classique : une phrase de 28 caractères tapée au hasard a environ une chance sur 10^{40} de sortir — à ce régime, des milliards de milliards de fois l'âge de l'univers n'y suffisent pas ; la même phrase, cherchée par variation *avec rétention de ce qui marche*, tombe en quelques dizaines de générations[3]. La rétention de ce qui marche : voilà la définition minimale de la méthode, et tout ce qui suit — le tri des demandes, le tableau, l'automatisation, l'enseignement — n'est que cette rétention, organisée. Sans elle, un projet ne capitalise rien : il rejoue indéfiniment le même tirage, et appelle ça de l'expérience.

Mais encore faut-il savoir ce qu'on appelle une méthode de travail, et je l'ai appris pendant mon projet de troisième année d'école d'ingénieur. Je travaillais pour une start-up qui réalisait le contrôle thermique de caves à vin de toutes tailles, du simple frigo au placard complet. Un frigo n'a pas une température homogène dans son enceinte — l'organe de refroidissement est quelque part, pas partout — et le problème s'aggrave avec la

taille. Le projet consistait à évaluer, puis à concevoir, des méthodes de simulation numérique capables de reconstruire la cartographie complète des températures dans l'enceinte, à partir de modèles thermodynamiques et de quelques sondes — en temps réel, sur un micro-contrôleur limité en puissance de calcul. Il fallait donc piloter le rapport entre précision et coût de calcul, ce qu'une simulation ordinaire sur station de calcul n'a jamais à faire. Le professeur qui m'encadrait n'avait pas plus de réponse que moi : le problème était assez singulier. C'est en le regardant dérouler la méthode devant moi, à froid et sans préparation, que j'ai compris la différence entre lui et moi. Aucun de nous deux n'avait la solution, mais lui savait par quel bout attaquer. « Commence par essayer A. Si ça marche, tu raffines avec B. Si ça ne marche pas, regarde C. Sur Google Scholar, le mot-clé pour ton problème, c'est X — regarde aussi Y. » En un quart d'heure, j'avais l'organigramme des solutions à tester, dans quel ordre, où les chercher, et une position de repli pour chaque tentative. Je n'avais toujours aucune solution ; j'avais un plan de bataille pour y arriver. Voilà ce qu'est une méthode. Que la solution A fonctionne ou non, j'apprendrais dans tous les cas quelque chose sur elle et quelque chose sur mon problème, et je savais déjà quoi faire à l'étape suivante : chaque essai devenait une expérience au sens propre — falsifiable, instructive dans ses deux issues. Sans méthode, j'aurais péniblement lancé des recherches avec les mauvais mots-clés, bricolé du code, et réinventé de travers des modèles réduits que la littérature avait déjà éprouvés. Et même l'échec complet de l'arbre entier aurait dit quelque chose de précis : que le problème était mal posé dès le départ, dans les mauvais termes ou avec les mauvais critères — le chapitre « Poser

le problème » a montré ce que vaut ce diagnostic-là. Avec ou sans solution, quoi qu'il arrive, je finissais avec une meilleure compréhension du problème. Le savoir du professeur n'était pas la réponse : c'était l'élagage de l'espace des solutions — un quart d'heure de structure du problème contre d'essais aléatoires à l'aveuglette.

Cette leçon vient d'un projet de recherche, mais elle se transfère à n'importe quel travail, dès qu'on accepte de faire du travail lui-même un sujet de recherche : chercher la meilleure façon d'accomplir la tâche, par des essais qui enseignent quelque chose dans tous les cas, pour travailler mieux, et moins, la fois suivante. Or pour beaucoup de gens, l'expérience se limite à reproduire les mêmes gestes sans variation et sans les questionner pendant des décennies, ce qui caractérise une expérience dépourvue d'expérimentation, autrement dit : une habitude qu'on ne peut pas expliquer.

Pendant des années, j'ai porté mon logiciel à peu près seul — développement, publication, assistance, documentation. À cette échelle, on ne tient pas en travaillant plus : on tient en travaillant *pour le travail*, c'est-à-dire en consacrant une part des heures non pas à la tâche du jour, mais à l'outillage, aux méthodes et à l'organisation qui rendront la tâche de demain plus fiable, puis inutile. Chaque heure racontée ici n'a pas résolu un problème : elle a supprimé une catégorie de problèmes. C'est l'exact inverse du régime décrit au chapitre « Le travailleur invisible », où plus on avance, plus on casse.

Tout commence à l'entrée, parce que c'est par l'entrée que tout déborde : les demandes. La règle de mon projet est brutale, et elle découle du chapitre « Écouter sans obéir » : *les demandes de fonctionnalités ne sont pas reçues au fil de l'eau.* C'est le concepteur qui consulte les utilisateurs sur leurs besoins, au moment où un chantier de conception s'ouvre — et non les utilisateurs qui saisissent le concepteur au gré de leurs envies. Il y a à cela deux raisons. La première est cognitive : une demande qui tombe au milieu d'un chantier est une interruption, et l'on verra plus bas ce qu'une interruption coûte réellement. La seconde est de méthode, et le chapitre « Écouter sans obéir » l'a établie : une demande formulée en solution — « ajoutez tel bouton » — est irrecevable ; seule compte la demande formulée en besoin, situé dans le flux de travail d'un métier — « je dois pouvoir accomplir telle tâche ». Filtrer à l'entrée n'est pas mépriser l'utilisateur : c'est refuser de bâtir la créature de Frankenstein une greffe à la fois. Les demandes qui n'entrent dans aucun chantier ouvert attendent qu'un chantier s'ouvre ; celles qui ne sont que des solutions déguisées sont retraduites en besoins, ou refusées.

Vient ensuite l'organisation du travail lui-même, et elle tient en peu de règles, toutes écrites. Le travail en cours est affiché sur un tableau *kanban* : à faire, en cours, à tester. L'outil vient de l'industrie — chez Toyota, le kanban est né comme système de signaux visuels servant précisément à limiter le travail entamé et à tirer la production par l'aval plutôt que la pousser par l'amont[4] — et sa transposition garde l'essentiel : rendre l'encours visible, donc limitable. Chez moi, la limite est de deux

chantiers de programmation ouverts en même temps, jamais plus, et choisis *indépendants* l'un de l'autre : on bascule sur le second pendant que le premier attend des retours d'usage, ce qui occupe l'esprit sans l'éparpiller ; et comme les deux chantiers ne se touchent pas, on sait toujours lequel a introduit quelle régression. C'est le contraire de la ruche du chapitre « Le comité n'inventera rien » : peu de fronts, tenus l'un après l'autre. Enfin, au milieu d'un chantier, je ne traite pas les sujets qui lui sont étrangers, et je ne les écoute pas non plus. Ce n'est pas un caprice : la psychologie du travail a mesuré ce que coûte l'interruption — le travail interrompu se rattrape en travaillant plus vite, mais au prix d'un stress, d'une pression et d'un effort nettement supérieurs, et la simple reprise d'une tâche complexe abandonnée demande de longues minutes de réimmersion[5]. La programmation est un travail qui n'existe que par blocs : un bloc brisé n'est pas à moitié fait, il est à refaire. Cette discipline paraît rigide ; elle est ce qui permet à un homme seul de porter la charge d'une équipe.

L'étage suivant est l'automatisation, et c'est le cœur du « travailler pour le travail » : toute corvée répétée à la main est une dette qu'on peut rembourser une fois pour toutes. Les paquets d'installation pour Windows et Linux se compilent tout seuls, chaque nuit, à la seule condition que du code ait été poussé la veille, et leurs liens se publient dans un canal dédié : n'importe quel utilisateur peut essayer, le soir même, ce qui a été écrit dans la journée, sans que personne n'ait rien compilé ni publié à la main. L'idée n'est pas neuve — la compilation quotidienne avec test de fumée est une des plus vieilles bonnes pratiques

documentées du génie logiciel, et l'intégration continue en est l'héritière[6] — mais le chapitre « Livrer, pas larguer » en faisait une exigence de test ; ici, c'est une machine qui tourne sans moi, et c'est toute la différence. Même logique pour le débruiteur du chapitre « Finir et maintenir » : il n'a pas été livré comme un réseau de neurones figé, mais comme une chaîne — tri des images côté utilisateurs, validation côté mainteneur, réentraînement automatisé — de sorte qu'améliorer le modèle l'an prochain ne redemandera pas le travail de cette année. À chaque fois, la question n'est pas « comment faire cette tâche ? », mais « que faut-il construire pour que cette tâche, la prochaine fois, se fasse sans moi ? ».

Le cas extrême de cette logique s'attaque à la source d'épuisement la plus sournoise du chapitre « Le travailleur invisible » : la communication répétitive. Devenu le visage public du projet, je passais un temps déraisonnable à répondre à des questions dont la réponse existait déjà, écrite par moi, quelque part sur l'un de mes sites : la documentation se perdait dans les limbes d'Internet, et le mainteneur servait de moteur de recherche en chair et en os. Sur les forums, des bénévoles de bonne volonté s'usent à la même tâche : répéter la même information, recoller les mêmes liens, pour des vagues sans fin de débutants. Or ces débutants ne sont pas paresseux ; ils butent sur un problème que l'ergonomie a mesuré il y a 40 ans sous le nom de *problème du vocabulaire* : la probabilité que deux personnes désignent spontanément la même chose par le même mot est inférieure à 20 %[7]. Un moteur de recherche classique exige de connaître les mots-clés ; le débutant, par définition, ignore le jargon qui lui

donnerait son point d'entrée dans le système d'information. J'ai donc construit l'outil qui manquait. D'abord un secrétaire virtuel qui trie le courrier entrant ; puis, par-dessus, un moteur de recherche sémantique — je l'ai appelé Chantal — qui apprend les parentés de sens entre les mots, retrouve une réponse par le sens plutôt que par le mot (symbole) exact, et surtout *propose* à l'utilisateur les mots-clés voisins de sa question : il lui rend l'entrée du labyrinthe. Entraîné sur près d'un million de pages tirées de mes sites, de la documentation et de sources fiables, indexant 400 000 pages en deux langues, il répond avant qu'on ait besoin de me déranger. La leçon technique vaut d'être notée, car elle résume le livre : 80 % de l'effort n'est pas allé au modèle d'apprentissage, mais au nettoyage des données — des ordures en entrée font des ordures en sortie —, et un modèle réglé trop finement sur son corpus fait un plus mauvais moteur de recherche, pas un meilleur. L'essentiel du travail fut, une fois de plus, invisible et ingrat ; il se rembourse à chaque question à laquelle je n'ai plus à répondre.

Et l'outil a suivi la pente naturelle de cette logique, qui est la vraie leçon du chapitre : il a cessé d'être à moi. Conçu d'abord pour délester le support — retrouver les pages de documentation que les mêmes questions redemandaient sans fin —, le moteur a ensuite avalé la documentation de développement, puis les ressources du métier lui-même, les spécifications OpenMP et OpenCL entre autres, puis l'intégralité des commits, des propositions de code et des *issues* du dépôt — puis la littérature académique du domaine : capteurs, science de la couleur, sociologie de l'image. Ce qui était mon pare-feu est devenu

l'infrastructure de connaissance de tout le monde — non plus un index de logiciel, mais un portail de spécialité sur l'imagerie, ancré dans un projet logiciel : l'utilisateur y trouve son mode d'emploi, le contributeur d'aujourd'hui ses spécifications et ses interfaces, celui de demain l'historique des décisions qui l'ont précédé, et le curieux la science qui fonde le tout. Il faut donc énoncer « travailler pour le travail » dans toute son extension : on ne travaille pas seulement pour son moi futur — on travaille pour ses collègues présents et futurs, pour quiconque devra un jour faire tourner la machine. Un flux de travail bien construit ne rend pas seulement service à son auteur : il abaisse le coût du travail pour tous ceux qui le partagent, y compris ceux qui ne sont pas encore arrivés. Et j'irai au bout de l'idée : c'est la forme la plus profonde du respect des personnes, utilisateurs et contributeurs confondus. Pas le respect des politesses, mais le respect en actes : rendre le travail de chacun plus facile, et rendre chacun capable de le faire sans soi. L'autonomie d'autrui est le seul hommage qui ne coûte rien à celui qui le reçoit et qui grandit avec le temps ; la conclusion de ce livre dira que ce respect-là porte un autre nom.

Reste la source de charge la plus mal comprise, et son traitement relève du chapitre « Il n'y a pas de problème technique » : une grande part de ce qui arrive déguisé en rapport de bug ou en demande de fonctionnalité est en réalité un défaut de formation. Un utilisateur mal formé a des attentes fausses ; prendre ses demandes au pied de la lettre, c'est charger le logiciel de fonctions qui dupliquent l'existant et alourdissent la machine pour rien. Ce problème-là ne se corrige pas dans le code : il se corrige à la racine, en enseignant. D'où un investissement que peu de projets

consentent : une documentation qui suit l'ordre d'apparition des fonctions dans l'interface, des tutoriels en vidéo, des guides organisés par tâche de métier plutôt que par module, et des ressources théoriques pour que l'utilisateur apprenne à diagnostiquer lui-même. Enseigner est encore du « travailler pour le travail » : chaque page qui apprend à un utilisateur à se dépanner seul tarit une source de sollicitations. La tranquillité du travailleur du chapitre « Le travailleur invisible » ne s'obtient pas seulement en se protégeant ; elle s'obtient en rendant l'utilisateur autonome — c'est le même geste, vu de l'autre bout.

L'objection est prévisible, et je l'ai entendue : tout ce temps passé à outiller, automatiser, documenter, filtrer, c'est autant de temps qui n'est pas passé à écrire le logiciel — du méta-travail, un luxe que les projets débordés ne peuvent pas se payer. La réponse est arithmétique, et elle rend tangible le pari du chapitre « Le travailleur invisible » : l'outillage est un placement à intérêts composés. L'heure investie dans la compilation nocturne se rembourse chaque nuit ; l'heure investie dans Chantal, à chaque question évitée ; l'heure investie dans le tri des demandes, à chaque créature de Frankenstein non construite. Refuser d'outiller, c'est choisir de payer éternellement plein tarif pour chaque tâche répétée — et c'est très exactement ainsi qu'un mainteneur s'épuise : non par excès de vrai travail, mais par la répétition d'un travail qu'il aurait pu supprimer. Au « on verra quand on aura le temps », ce chapitre répond : on n'a jamais le temps *parce qu'*on n'outille pas. La boucle ne se casse qu'en travaillant, un moment, pour le travail plutôt que pour la tâche.

Le rendement de cette organisation se mesure — à une échelle qui n'est pas comparable à celle des chapitres « Livrer, pas larguer » et « Le comité n'inventera rien » : Ansel reçoit 10 fois moins de signalements que Darktable. Mais c'est précisément le régime qui se compare, pas le volume. Sur les 717 *issues* reçues depuis 2022, 92 % sont fermées — contre à peine plus de la moitié dans les cohortes récentes de Darktable — et pas une seule par un robot : le projet n'a pas d'automate de péremption, chaque fermeture est un acte humain. Le tri existe : régression, priorité en 4 niveaux, trace de plantage jointe, « pas clair » — la taxonomie du triage couvre l'essentiel du registre, là où 44 % des *issues* récentes de Darktable ne portent aucune étiquette. La médiane de fermeture est plus lente — 7 jours contre un et demi — et c'est le prix du régime : Darktable ferme vite ce qu'il ferme et abandonne le reste à l'automate ; ici l'on ferme presque tout, donc parfois tard — un dixième des cas attend plus d'un an. Et sur 2026, année de pointe avec 300 *issues* reçues, il s'en est fermé davantage qu'il n'en est arrivé : l'encours est tombé à deux. Ce n'est pas une vertu : c'est l'arithmétique d'une entrée filtrée, d'un tri fait à l'arrivée et d'un refus de l'automate. Le même flux d'*issues*, organisé, cesse d'être un torrent qu'on subit[8].

Mais il faut dire pourquoi cette boucle ne se casse presque jamais dans le logiciel libre, car la raison est structurelle, et elle porte plus loin que l'outillage. Le contributeur bénévole vient le soir, après sa journée de travail, et il vient pour une chose précise : la joie de construire — le chapitre « Finir et maintenir » a décrit cette euphorie, et le chapitre « Ce que l'entre-soi produit » chiffrera ce que les enquêtes confirment : le premier motif déclaré de

contribution est la stimulation intellectuelle, le jeu. Or trier des demandes, écrire des règles, entretenir une chaîne de compilation, nettoyer des données, documenter : c'est précisément à quoi ressemble sa journée de travail — du processus, de l'administration, des tâches ingrates. Il l'a fuie. Demander à un bénévole de faire le soir, gratuitement, ce qu'il est payé pour endurer le jour, c'est demander à l'eau de remonter la pente. Et ce n'est pas moi qui le leur fais dire : dès 2016, sur le forum de la communauté, à un utilisateur qui demandait qu'on stabilise l'existant plutôt que d'empiler les projets, un contributeur répondait que « l'immense majorité d'entre nous n'est pas payée » et qu'il est « déplacé de dire à quelqu'un de travailler sur quelque chose de plus utile » — le loisir revendiqué comme clause d'exemption, noir sur blanc[9]. Personne ne le fait, et ce n'est pas de la paresse : c'est la construction même du modèle. Un système qui recrute sur le plaisir obtient exactement ce que le plaisir finance — des fonctionnalités neuves, des prototypes excitants — et rien de ce qui rend un outil robuste, car la robustesse est justement la part du travail qui n'amuse pas. C'est mon diagnostic le plus dur, et ce chapitre le rend inévitable : le modèle de contribution du logiciel libre, fondé sur le bénévolat, est structurellement condamné — non par manque de talent ni de bonne volonté, mais parce qu'il sélectionne par construction le travail agréable et laisse orphelin le travail qui compte.

Le modèle échoue d'ailleurs jusque sur son propre terrain. Darktable a énoncé sa règle en conférence, au mot près : s'il y a une règle, c'est de ne jamais empêcher quelqu'un de coder tant qu'il s'amuse[10] — le chapitre « Le cercle fermé » reviendra sur

cette doctrine. Elle promettait, par le plaisir, un afflux perpétuel de bras. Or les séries du chapitre « Le travailleur invisible » disent l'inverse : commits, propositions et relectures en recul de deux tiers, cœur travaillant passé de 27 à 13 personnes entre 2020 et 2025. Ma théorie est que le code lui-même est devenu le repoussoir : des années de fonctionnalités fusionnées sans architecture — le chapitre « Le culte du cargo » a fait l'arithmétique de ces fusions sans relecture — font un plat de spaghettis que personne n'a envie de démêler. Je l'ai vérifiée sans le vouloir : j'ai entrepris de recruter deux développeurs chevronnés de ma connaissance ; ils ont ouvert le code, et ils ont dit non merci. Programmer cesse d'être un plaisir quand le programme est illisible. La règle unique s'est donc battue elle-même : à force de protéger le plaisir immédiat de chacun contre toute discipline, elle a fabriqué un code qui ne fait plus envie à personne, et le recrutement se tarit précisément là où tout était organisé pour lui. On n'en sort pas par l'exhortation ; on n'en sort que par une structure qui paie, ou mandate, le travail ingrat. C'est ce que la dernière partie construira.

Ce chapitre s'achève ainsi sur une note qui tempère son propre diagnostic, et c'est voulu : le collectif du logiciel libre échoue souvent, mais il n'y est pas condamné. Le comité n'invente rien — un responsable outillé, si ; le travailleur invisible est broyé — un projet qui capitalise le protège. Mais toutes ces règles, le tri, le tableau, l'automatisation, l'enseignement, supposent quelqu'un qui ait le mandat de les écrire et de les faire tenir. On retombe sur la question que ce livre ne cesse de croiser : de quel droit, dans quelle structure, avec quelle légitimité ? Les parties précédentes

ont montré ce qu'il en coûte de ne pas répondre. Avant d'y venir, il reste une question que ce chapitre a laissée intacte : de tout ce travail qu'on vient d'organiser, lequel fallait-il accepter ? C'est l'objet du chapitre suivant.

---

1. Wolpert, D. H. & Macready, W. G. (1997). « No Free Lunch Theorems for Optimization ». *IEEE Transactions on Evolutionary Computation*, 1(1), 67-82 https://doi.org/10.1109/4235.585893 — « the average performance of any pair of algorithms across all possible problems is identical », y compris quand l'un des deux est le tirage au hasard ; et pour qu'une stratégie en batte une autre, la structure du problème « must be known and reflected directly in the choice of algorithm ». ↩

2. Simon, H. A. (1996). *The Sciences of the Artificial*, 3e éd. Cambridge, MA : MIT Press, chap. 5 (« The Science of Design : Creating the Artificial ») https://mitpress.mit.edu/9780262691918/the-sciences-of-the-artificial/ : la conception comme recherche sélective dans un espace de possibles, l'allocation des ressources de recherche, et le « satisficing » comme critère d'arrêt — on cherche une solution suffisante, non optimale, parce que l'espace excède toute exploration exhaustive. ↩

3. Dawkins, R. (1986). *The Blind Watchmaker*. Harlow : Longman, chap. 3 (« Accumulating Small Change ») ; trad. fr. *L'Horloger aveugle*, trad. B. Sigaud, Paris : Robert Laffont, 1989. La phrase-cible METHINKS IT IS LIKE A WEASEL —

28 caractères sur un alphabet de 27 signes — a une chance sur $27^{28} \approx 10^{40}$ d'être tapée d'un coup ; Dawkins estime la sélection « en une seule étape » à environ 10^{30} années, plus d'un milliard de milliards de fois l'âge de l'univers. La sélection *cumulative* — variation aléatoire, mais rétention du meilleur candidat à chaque génération — atteint la même cible en 43, 64 et 41 générations sur ses trois essais. ↩

4. Ohno, T. (1988). *Toyota Production System : Beyond Large-Scale Production*. Cambridge, MA : Productivity Press, chap. 2 https://doi.org/10.4324/9780429273018 (déjà cité au chapitre « La racine et le symptôme » ; trad. fr. *L'Esprit Toyota*, Paris : Masson, 1989) — le kanban, « idée que j'ai prise aux supermarchés américains », est l'outil de pilotage du système : sa première règle est que le poste aval vient chercher au poste amont, et sa fonction est d'empêcher la surproduction — le nombre de cartes en circulation plafonne l'encours, et le contrôle est visuel, « une gestion par le regard ». ↩

5. Mark, G., Gudith, D. & Klocke, U. (2008). « The Cost of Interrupted Work : More Speed and Stress ». *Proceedings of CHI 2008*, ACM, 107-110 https://doi.org/10.1145/1357054.1357072 — le travail interrompu est rattrapé en accélérant, au prix d'un stress, d'une frustration et d'un effort significativement plus élevés. Sur la reprise : Mark, G., González, V. M. & Harris, J. (2005). « No Task Left Behind ? Examining the Nature of Fragmented Work ». *Proceedings of CHI 2005*, ACM, 321-330 https://doi.org/10.1145/1054972.1055017 — quand une tâche

interrompue est reprise le jour même, elle ne l'est en moyenne qu'au bout de 25 minutes, après passage par deux autres sujets (le « 23 minutes » qui circule dans la presse ne figure dans aucun article publié). Sur l'immersion : DeMarco, T. & Lister, T. (1987). *Peopleware : Productive Projects and Teams*. New York : Dorset House, chap. 10 https://openlibrary.org/books/OL2392791M/Peopleware (trad. fr. de la 1re éd. : *Les Hommes de l'ordinateur*, Paris : Masson, 1991) — entrer dans l'état de concentration productive « demande quinze minutes ou plus », d'où leur facteur E : les heures ininterrompues, non les heures de présence, mesurent le travail réel. ↩

6. McConnell, S. (1996). « Daily Build and Smoke Test ». *IEEE Software*, 13(4), 143-144 https://doi.org/10.1109/MS.1996.10017 — « la compilation quotidienne avec test de fumée maintient les erreurs d'intégration petites et gérables » ; prolongée par Fowler, M. « Continuous Integration ». martinfowler.com, 2000, réécrit en 2006 https://martinfowler.com/articles/continuousIntegration.html. ↩

7. Furnas, G. W., Landauer, T. K., Gomez, L. M. & Dumais, S. T. (1987). « The Vocabulary Problem in Human-System Communication ». *Communications of the ACM*, 30(11), 964-971 https://doi.org/10.1145/32206.32212 — « dans tous les cas, deux personnes favorisaient le même terme avec une probabilité inférieure à 0,20 » ; un accès conçu sur le vocabulaire d'un seul concepteur échoue donc « dans 80 à 90

% des situations courantes ». Leur remède, l'« aliasing illimité », est l'ancêtre de la recherche par le sens décrite ici. ↩

8. Dépôt aurelienpierreeng/ansel https://github.com/aurelienpierreeng/ansel, arrêté au 23 juillet 2026 : 717 *issues* créées depuis 2022 (36, 159, 54, 160 puis 308 par an), 658 fermées (92 %), 59 ouvertes ; en 2026 : 308 créées, 339 fermées. Médiane de clôture 7,4 jours, 75e centile 99 jours, 90e centile 399 jours. Aucune étiquette de péremption automatique n'existe dans le registre ; étiquettes principales : *regression* (83), priorités (critique 20, haute 78, moyenne 63, basse 54), *backtrace: included* (48), *unclear* (33). 138 rapporteurs distincts, dont 32 à 3 *issues* ou plus ; 8 % des *issues* sont déposées par le mainteneur lui-même (le tableau kanban décrit plus haut passe par le traqueur). 19 % des *issues* n'ont aucun commentaire — part pour l'essentiel fermée par le correctif lui-même, le registre d'un projet à deux ne conversant pas pour converser. ↩

9. https://discuss.pixls.us/t/975, mars 2016 (archivé ; le fil est analysé au chapitre « Ce que l'entre-soi produit ») : « the vast majority of us are not getting paid to hack on open source software. As long as people are doing all of this in their spare time I think it's inappropriate to tell them to work on something more useful » (notre traduction). ↩

10. Au mot près : « if there is a rule, it's don't stop anyone from coding when they're having fun » — diapositive « do-ocracy » de la présentation officielle du projet au Libre Graphics Meeting 2019 (vidéo, 17 min 31

https://www.youtube.com/watch?v=o4cnoNMdZBw), dont le chapitre « Le cercle fermé » analyse les trois diapositives clés — la règle, la responsabilité, et le bilan qui s'achève sur « technical debt :/ ». ↩

23. L'ambition non héroïque

Il y a deux façons d'être ambitieux pour un logiciel. La première se voit : une fonctionnalité spectaculaire, annoncée, applaudie le jour de la sortie. La seconde ne se voit pas : le logiciel marche encore dans 10 ans, et personne n'a eu à s'en apercevoir. La première est héroïque. La seconde est un héroïsme ordinaire, silencieusement utile, discrètement efficace.

Le chapitre « Finir et maintenir » a posé la règle : ne rien entreprendre qu'on n'ait les moyens de finir et de maintenir. Il laissait ouvertes deux questions, et ce sont celles qui comptent en pratique. Au moment d'accepter une fonctionnalité, comment sait-on ce qu'elle coûtera à tenir ? Et qui a le pouvoir de dire non ? Ce chapitre répond avec le dossier Darktable en main, pour que la réponse serve à ceux qui refusent de finir pareil.

Il faut d'abord retirer à la maintenance son statut de sanction. On en parle comme du prix d'un code mal écrit : bien conçu, le logiciel n'en réclamerait pas. C'est faux, et la question est tranchée depuis un demi-siècle. Meir Lehman a formulé, entre 1974 et 1996, 8 lois empiriques de l'évolution des logiciels ; les 3 mieux validées disent qu'un programme réellement utilisé doit changer sans cesse, que sa complexité augmente, et que sa qualité perçue se dégrade[1]. La clause qui accompagne la troisième importe plus que la loi elle-même : la qualité se dégrade *à moins qu'un travail spécifique ne soit investi pour l'en empêcher.*

Autrement dit, la dégradation est l'état par défaut. Un logiciel qu'on cesse d'entretenir ne reste pas où il est : il se défait sur place, parce que tout bouge autour de lui — le système d'exploitation, les bibliothèques dont il dépend, les formats de fichiers, les machines, les usages. Même dans un cas idéal, la maintenance n'est pas le remboursement d'une faute passée : c'est le loyer de l'existence. Et un loyer se budgète d'avance. Évidemment, si le prix des fautes passées s'ajoute, le budget explose.

Reste à savoir combien. La réponse est connue, stable, et régulièrement ignorée : de 60 à 80 % du coût total d'un logiciel sur son cycle de vie est postérieur à la livraison. L'étude fondatrice de Lientz et Swanson, menée sur près de 500 organisations, en donne la répartition : la maintenance dite *perfective* — l'adaptation à des besoins qui ont changé — pèse à elle seule plus de la moitié de l'effort, et les trois quarts du travail de maintenance ne consistent pas à réparer des pannes mais à suivre le mouvement du monde[2]. Ces proportions ont tenu 40 ans, à travers tous les changements de technologie.

La conséquence est brutale pour la façon dont on décide. Quand une équipe discute d'une fonctionnalité, elle discute de ce qu'elle coûtera à écrire. Or ce coût-là représente le quart ou le tiers de la dépense réelle. Le reste viendra plus tard, sera payé par quelqu'un d'autre, et n'aura jamais figuré au devis. Décider une fonctionnalité sur son coût de développement, c'est en débattre le prix en n'en connaissant que le quart.

Si la maintenance est à ce point la loi du milieu et l'essentiel de la dépense, il reste à expliquer pourquoi elle est méprisée. La réponse n'est pas technique : elle tient à ce que ce travail *donne à voir*. Il est répétitif. Il ne produit rien de neuf. Il ne s'annonce pas — personne ne publie « cette semaine, le logiciel a continué de fonctionner ». Il consiste à revenir sur l'ouvrage d'un autre, à comprendre avant de toucher, à remettre en état ce qui existait déjà. Ces propriétés désignent, dans la division ordinaire du travail, une catégorie précise de tâches : celles qu'on délègue. Aux femmes, pour ce qui relève du foyer ; aux subalternes, au personnel dit de support, aux moins qualifiés, pour ce qui relève de l'entreprise. Le chapitre « Ce que l'entre-soi produit » en a cité la formulation la plus nette : l'obsession de la nouveauté technologique masque tout le travail — le travail domestique compris — que les femmes accomplissent de façon disproportionnée.

Il faut retenir ce déplacement, car il porte plus loin que la maintenance : un travail universellement requis, historiquement assigné aux subalternes, et dévalué pour cette raison même, c'est très exactement la définition que l'éthique du soin donne du soin.

Or ce classement est faux, et il est faux d'une manière qu'on peut établir. Le chapitre « Le code est un passif » a donné la règle de Kernighan et Plauger : déboguer est deux fois plus difficile qu'écrire le code, de sorte que celui qui écrit le code le plus astucieux dont il est capable est, par construction, trop peu astucieux pour le déboguer. Le travail qu'on relègue est le plus dur des deux. La division n'est donc pas une hiérarchie de difficulté : c'est une hiérarchie de visibilité, et elle classe à

l'envers. Les meilleurs programmeurs devraient être employés à la maintenance. Mais faire partie des meilleurs ouvre des possibilités plus prestigieuses.

Le mécanisme de cette invisibilité a été décrit ailleurs, et il est cruel. Susan Leigh Star a montré qu'une infrastructure ne se remarque qu'en tombant en panne, et que le travail qui la tient debout appartient à une catégorie que les organisations savent produire mais pas compter : le travail invisible — invisible non par accident, mais parce que les systèmes de description eux-mêmes rendent certaines tâches comptables et d'autres non[3]. La maintenance réussie s'efface d'elle-même. Bien faite, elle ne laisse rien à montrer ; n'ayant rien laissé à montrer, elle ne peut pas être portée au crédit de qui l'a faite. Le mainteneur n'a qu'une seule façon de prouver son travail : cesser de le faire. On ne voit pas un bug réglé avant qu'il en devienne un.

Cela n'est pas sans rappeler le topos du super-héros dans la culture populaire, qui sauve le monde chaque semaine, tout seul dans un coup d'éclat fracassant. On n'y montre jamais ceux qui déblaient les gravats, rouvrent l'hôpital, rétablissent l'eau et relèvent les lignes électriques — le travail qui occupe pourtant les mois suivant chaque catastrophe, et sans lequel la ville sauvée reste inhabitable. La morale implicite du genre ne varie pas : ce qui mérite d'être raconté est l'exploit ponctuel d'un individu exceptionnel, jamais l'entretien continu d'un collectif ordinaire. Le super-héros de comics a son pendant dans la mythologie professionnelle.

Reste à savoir si cette répartition du prestige est un accident. Je ne le crois pas, et voici sur quoi je m'appuie. La composition du milieu est mesurée, et le chapitre « Ce que l'entre-soi produit » la donne : des hommes, la cinquantaine, de formation d'ingénieur, et 2 % de femmes. Ce que ces gens font ici, ils le font bénévolement, le soir, sans que personne ne le leur assigne : leurs choix ne révèlent donc pas des contraintes, ils révèlent des dispositions. C'est ce que Bourdieu nomme l'habitus — des schèmes acquis ailleurs, dans un métier et une position sociale, et rejoués ici sans qu'on ait à les décider. Il est facile d'imaginer que dans un milieu dont les membres occupent le plus souvent, dans leur emploi rémunéré, le versant conception du travail — celui où l'entretien est fait par d'autres —, la division reconduite bénévolement est celle qu'ils connaissent déjà. Concevoir et annoncer d'un côté ; entretenir de l'autre, c'est-à-dire dans l'ombre.

Cette division n'est pas une déduction sociologique : le dépôt la chiffre, commit par commit[4]. Depuis 2019, 252 personnes ont modifié le code. Dans l'ordre des questions — qui répare, combien, depuis quand :

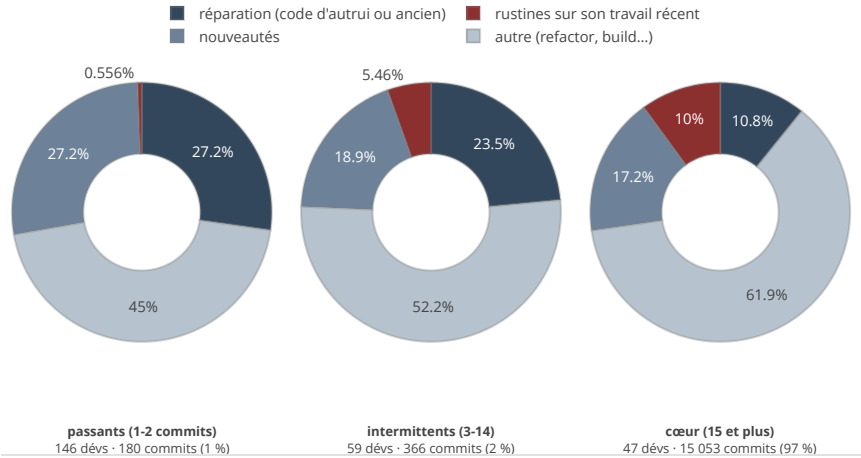
- **Qui n'a jamais réparé** : 48 % des développeurs n'ont signé aucun correctif depuis 2019 ; en ajoutant ceux dont les correctifs ne rapiéçaient que leur propre travail des semaines précédentes, la moitié n'a jamais réparé quoi que ce soit qu'elle n'ait pas elle-même cassé.
- **La participation suit l'engagement** : parmi les passants à 1 ou 2 commits, 32 % ont réparé au moins une fois ; parmi les

intermittents (3 à 14 commits), 56 % ; dans le cœur (15 commits et plus), 96 %.

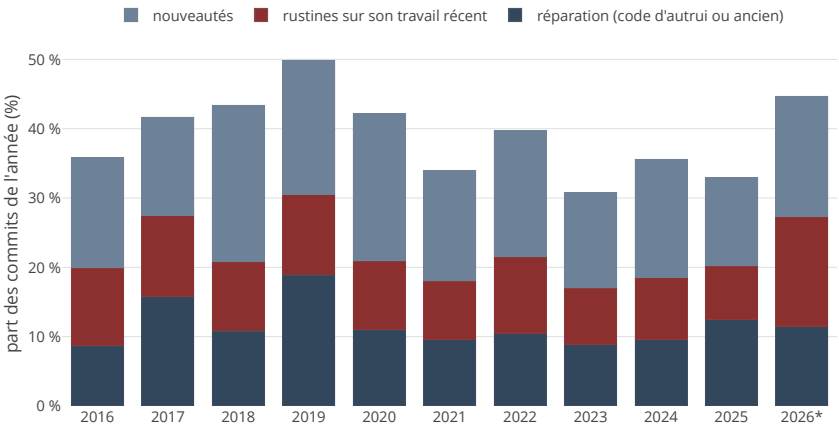
- **Le volume est l'affaire du cœur** : 47 personnes signent 97 % des commits et 92 % de la vraie réparation. Les 146 passants pèsent 1 % du flux — leur rare contribution est d'ailleurs souvent un correctif, mais elle ne pèse rien en masse.
- **La moitié des « correctifs » n'en sont pas** : sur 3 289 commits déclarés correctifs, 1 531 rapiècent le propre travail récent de leur auteur — la taxe des nouveautés, pas de la maintenance. Cette rustine est à 99 % un phénomène du cœur, mécaniquement : il faut un travail récent à rapiécer.
- **La réparation vraie — le code d'autrui, le code ancien — pèse 1 commit sur 9** (1 758 sur 15 599), et ce poids est stable depuis 2016 : entre 9 et 19 % selon l'année, sans tendance. Aucune époque du projet, même à 83 développeurs actifs, n'a donné davantage à l'entretien.
- **Ce qui s'effondre est le volume, pas la proportion** : environ 350 commits de réparation vraie par an au pic 2019-2020, 128 en 2025 (−60 %) — au rythme, exactement, de la fonte du cœur.

Et il faut rapporter ce dernier chiffre à la littérature sur laquelle ce chapitre s'est ouvert : 60 à 80 % du coût d'un logiciel sur son cycle de vie est postérieur à la livraison. Darktable est en service depuis dix-sept ans — il est tout entier dans cette phase, celle où, selon l'industrie, l'entretien domine le budget. Il y consacre un commit sur neuf, soit 11 %, et continue d'embarquer des nouveautés chaque année. La « maintenance » de Lientz et

Swanson englobe, il est vrai, plus que la correction — l'adaptation, le perfectionnement — et un commit n'est pas un euro ; mais l'écart n'est pas une nuance de définition, c'est un ordre de grandeur. Un logiciel dont tout le coût restant est de l'entretien, et qui donne à la réparation le neuvième de son énergie pendant que les nouveautés continuent d'entrer, ne fait pas des économies : il accumule la dette que ce chapitre a chiffrée, et il la laisse à d'autres.



Qui répare ? Composition des commits de code par palier d'engagement, 2019-2026 : réparation du code d'autrui ou ancien, rustines sur son propre travail récent, nouveautés, et le solde (refactorisation, build, divers). Les paliers n'ont pas le même poids : les passants pèsent 1 % des commits, le cœur 97 % — et il signe 92 % de la réparation vraie. La rustine est un phénomène du cœur : elle suppose d'avoir un travail récent à rapiécer.



Le mélange ne change pas : part des commits de l'année classés réparation vraie, rustine sur son propre travail récent, et nouveauté, 2016-2026 (le solde — refactorisation, nettoyage, build — complète à 100 % ; 2026 est arrêté au 31 juillet). Les proportions sont stables sur dix ans : la réparation vraie oscille entre 9 et 19 % sans tendance, quel que soit l'effectif. Ce que ce graphe ne montre pas, c'est le volume : environ 350 commits de réparation vraie par an au pic 2019-2020, 128 en 2025 — la proportion tient, les bras disparaissent.

Voilà pour le mépris. Le dossier en offre maintenant un cas d'école. Entre 2020 et 2022, le projet que j'ai quitté a engagé plusieurs refontes ambitieuses : le système de raccourcis clavier, réécrit pour rendre chaque action pilotable depuis un contrôleur externe ; l'organisation des modules de traitement en groupes, transformée en outil de mise en page configurable ; les filtres de collection, c'est-à-dire les commandes qui trient la photothèque, refondus deux fois en 4 mois. Ce livre a déjà donné leurs comptes de lignes aux chapitres « Sous l'interface, les dragons » et « Le

comité n'inventera rien ». Ces 3 chantiers ont deux propriétés communes, et ce sont elles qui font le sujet de ce chapitre.

La première : aucun ne répondait à un problème préalablement établi. Ni cahier des charges, ni douleur mesurée, ni public identifié — le chapitre « Poser le problème » a dit ce qui manque quand tout cela manque. La seconde, plus grave : l'ambition s'est arrêtée à la livraison. Personne n'avait chiffré ce que ces refontes coûteraient à maintenir, parce que personne n'avait à le faire.

Le résultat se mesure aujourd'hui, et il prend la forme d'un régime à deux vitesses. Parmi les signalements de bugs qui finissent par être effectivement corrigés, la moitié se ferment en 1,3 jour ; mais le neuvième sur dix, dans l'ordre des délais, attend déjà plus de 22 jours, et 47 % de tous les signalements ne seront jamais réglés, mais fermés par le robot de péremption[5]. Ce n'est pas la distribution d'une équipe débordée qui traite les demandes dans l'ordre d'arrivée. C'est la signature de deux populations distinctes : d'un côté les corrections faciles, qu'on règle dans la journée ; de l'autre celles qui exigeraient de toucher à l'architecture, qui dépassent le temps et les compétences disponibles, et qu'on ne règle jamais. Entre les deux, presque rien. Un projet en bonne santé a une file d'attente ; celui-ci a un tri.

J'en ai fourni le témoignage à chaud : en 2022, pour la première fois en 4 ans de travail à plein temps, je ne parvenais plus à identifier la cause de la plupart des bugs d'interface, la logique de fonctionnement étant devenue opaque et le code incompréhensible[6]. Une architecture qu'on ne peut plus lire est une architecture qu'on ne peut plus réparer. Les fonctionnalités

ambitieuses de 2020-2022 n'avaient pas été enfermées dans des modules séparés : elles avaient été tissées à travers le reste du programme. Corriger l'une d'elles oblige à toucher tout ce qu'elle traverse, et chaque correction ajoute une interaction de plus. Le système ne se répare pas : il se complique en se réparant.

Resterait la sortie honnête, qui consisterait à retirer ce qu'on ne peut pas tenir. Elle est barrée par le mécanisme que le chapitre « L'éloge du simple » a décrit à propos des options : ce qui a coûté cher à construire paraît trop précieux pour être jeté, alors même que la dépense est faite et ne reviendra pas. On conserve donc une fonctionnalité à moitié cassée qu'on pourrait pourtant faire fonctionner, et l'on paie l'entretien du tout pour la seule moitié qui marche déniant l'autre.

Voilà pour le constat. La question difficile est en amont : au moment de dire oui ou non, comment évalue-t-on ce qu'une fonctionnalité coûtera pendant 10 ans ? Cette évaluation relève pour une bonne part de l'intuition du développeur expérimenté, et elle ne se démontre pas en réunion. C'est exactement le savoir *tacite* de Polanyi, invoqué au prologue : on sait plus qu'on ne peut dire, et l'ingénieur qui pressent qu'un chantier va mal tourner ne peut pas toujours exhiber la règle qui le lui dit. Cette intuition s'acquiert par l'exposition longue aux conséquences de ses propres décisions, c'est-à-dire en restant. D'où une conséquence que le chapitre « Le travailleur incompétent » développera sous un autre angle : le contributeur de passage, celui qui reste un an ou deux puis disparaît, ne peut pas l'avoir. Non qu'il soit moins doué : il partira avant d'avoir vu retomber les conséquences de ce qu'il a écrit, et propose donc de bonne foi

un travail dont le coût futur lui est structurellement invisible. Son départ emporte d'ailleurs le savoir avec lui — sur 1 932 projets libres étudiés, 16 % ont vu partir la totalité de leurs développeurs-clés, et le développement n'a été repris par d'autres que dans 41 % de ces cas[7].

Concéder la part d'intuition ne veut pas dire abdiquer, car une partie de cette évaluation est normalisée et une autre se mesure. Normalisée d'abord : la norme ISO/IEC 25010 ne traite pas la maintenabilité comme une vertu vague, elle la décompose en propriétés évaluables dont la première est la modularité — le degré auquel on peut modifier un composant sans affecter les autres. Viennent ensuite l'analysabilité, c'est-à-dire la capacité à diagnostiquer une panne et à évaluer l'impact d'un changement avant de le faire, puis la modifiabilité et la testabilité[8]. Ces mots fournissent une grille de questions à poser avant d'accepter : que faudra-t-il toucher pour changer ceci ? saura-t-on dire d'où vient une panne ? saura-t-on la tester ? Ce ne sont pas des questions de goût.

Mesurée ensuite, et c'est le point qu'on croit hors de portée. La « surface d'interaction avec le reste de l'application » n'est pas une image : c'est une grandeur. MacCormack, Baldwin et Rusnak l'ont formalisée sous le nom de coût de propagation — la proportion moyenne des éléments du système affectés lorsqu'on en modifie un pris au hasard. Elle se calcule sur une matrice de structure de conception, c'est-à-dire un tableau croisant tous les composants du programme pour marquer lesquels dépendent de quels autres. Ils l'ont appliquée à des logiciels réels, dont Linux et Mozilla : Linux se révèle nettement plus modulaire, et la grande

refonte de Mozilla a fait chuter son coût de propagation, ce qui donne à la mesure sa valeur de contrôle[9]. Un projet peut donc suivre sa propre modularité dans le temps comme il suit sa couverture de tests, et voir venir la dégradation au lieu de la découvrir.

Reste à dire pourquoi enfermer une fonctionnalité dans un module est le levier, et non une préférence d'esthète. La raison est ancienne : Parnas, cité au chapitre « Sous l'interface, les dragons », a montré qu'il faut découper un programme selon ce qui est susceptible de changer, afin qu'un changement reste local. Baldwin et Clark en ont donné la traduction économique : la modularité crée de la valeur d'option. Un module enclos peut être amélioré, remplacé ou supprimé sans toucher au reste — on achète ainsi le droit de changer d'avis plus tard. Un module diffus retire ce droit : il ne peut plus être ni remplacé ni retiré, seulement subi. La modularité n'est donc pas un raffinement qu'on ajoute quand on a le temps ; c'est elle qui décide si le projet aura encore des choix dans 5 ans.

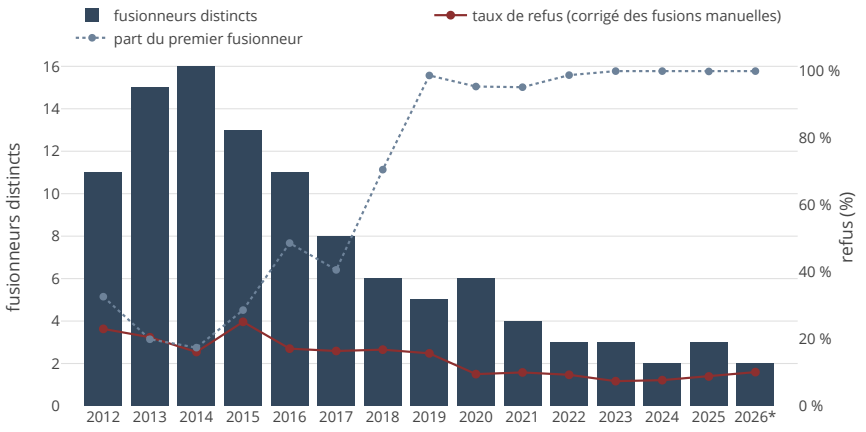
Une évaluation, cependant, ne sert à rien si personne n'a le pouvoir d'en tirer la conséquence. Dans le modèle de contribution par proposition de code, ce pouvoir existe et il a un titulaire : celui qui fusionne. Il peut refuser. La question est de savoir s'il refuse.

Le dossier répond, et la réponse est nette. J'ai compté, sur le dépôt du projet et pour chaque année, les propositions fusionnées et les propositions fermées sans avoir été fusionnées — donc le taux de refus parmi les seules propositions tranchées. Il s'établit

ainsi, une fois requalifiées en acceptations les propositions fermées sans fusion mais intégrées à la main (le commentaire de clôture en fait foi) :

- 23,0 % en 2012,
- 25,1 % en 2015,
- 16,8 % en 2018,
- 15,7 % en 2019,
- 10,0 % en 2021,
- 7,4 % en 2023,
- entre 8 et 10 % de 2024 à 2026[10].

Le taux de refus a été divisé par deux ou trois. Et la période où il s'effondre est exactement celle où le droit de fusionner se concentre : les pull requests étaient fusionnées par une quinzaine de personnes au début des années 2010, par deux ou trois depuis 2022 — un seul développeur signant chaque année entre 95 et 100 % des fusions depuis 2019 — la série complète est au chapitre « Penser contre soi ».



Le veto qui s'éteint, mesuré sur les pull requests (champ merged_by — la personne qui clique la fusion, quel que soit le mode), 2012-2026 : les barres donnent le nombre de personnes distinctes ayant fusionné des propositions dans l'année (échelle de gauche) ; la courbe pleine, la part des propositions réellement refusées — fermées sans fusion, déduction faite de celles intégrées à la main — rapportée aux seules propositions tranchées ; la courbe pointillée, la part du premier fusionneur (échelle de droite). Les échelles partent de zéro et aucune courbe n'est recalée sur une autre : la coïncidence des déclins est dans les données, pas dans le cadrage. 2026 est arrêté au 1er août.

On attendrait l'inverse. Un pouvoir qui se concentre est réputé trancher plus durement ; celui-ci tranche moins. Le veto n'a pas été exercé plus fermement par un seul homme : il a été exercé moins. Le chapitre « Le culte du cargo » a donné l'arithmétique qui l'explique — le volume fusionné par cet homme dépasse de

plusieurs fois ce qu'un être humain peut relire. On ne refuse pas ce qu'on n'a pas lu.

Le projet n'ignorait d'ailleurs pas le mécanisme. En juin 2019, à la conférence publique de ses 10 ans, le fondateur du projet et un des principaux développeurs de la première génération dressaient le bilan de ce qu'ils nommaient eux-mêmes leur politique de *laissez-faire* et en inscrivaient le 4e poste : de la dette technique — le chapitre « Le cercle fermé » cite la diapositive. Le diagnostic est donc contemporain de la dernière année haute de la courbe : 15,7 % de refus de propositions cette année-là, juste avant l'effondrement vers 9-10 %. La suite ne corrige pas le diagnostic : elle l'aggrave.

Encore faut-il rapporter ce taux à quelque chose, faute de quoi il ne dit rien. Le repère existe : sur le corpus de référence des projets hébergés sur GitHub, le taux de fusion moyen avoisine 72 %, et 13 % des propositions sont rejetées pour un motif technique[11]. Le projet refuse donc environ 2 à 3 fois moins que la moyenne du monde où il vit. Et c'est en le lisant avec ce qui entre que le chiffre devient accusateur. Un projet qui reçoit des contributions commandées — issues d'une feuille de route, discutées avant d'être écrites — peut légitimement les accepter presque toutes, puisque le tri a eu lieu en amont. Ce n'est pas le cas ici : les contributions arrivent non concertées, hors de tout programme, chacune portant l'idée que son auteur a eue dans son coin. Un flux que rien n'a trié en amont devrait être trié à l'entrée, donc davantage refusé, non moins. Un taux de refus de 9 % sur des propositions que personne n'a demandées ne signale

pas une communauté de contributeurs exceptionnellement justes : il signale une porte sans portier.

Il reste à regarder ce que ce refus de refuser dit de celui qui ne refuse pas — car les chiffres qui précèdent décrivent une conduite, non une intention. Le chapitre « Le conflit bien tempéré » rapporte l'aveu du mainteneur : il est parfois dur de prendre toutes les décisions seul, puis d'essuyer les reproches après coup. J'y montre ce que cette solitude devient lorsqu'on la raconte au lieu de la traiter. Il faut ici en nommer le mécanisme, car il est étudié.

Les sciences des organisations l'appellent le *romantisme du leadership*. Meindl, Ehrlich et Dukerich ont établi en 1985 que nous attribuons spontanément aux dirigeants la responsabilité des résultats d'une organisation, en négligeant les autres causes — l'équipe, le secteur, les circonstances. Ce n'est pas de la flagornerie, c'est un raccourci de mise en sens : expliquer une performance par ses causes réelles est coûteux et incertain, l'expliquer par un homme est immédiat[12]. Le raccourci fonctionne dans les deux sens, et c'est ce qu'on oublie : il sert aussi à celui qui occupe le centre. Si tout dépend de lui, alors l'état du projet est son mérite quand il va bien, et son fardeau quand il va mal — jamais une propriété de la structure.

On peut alors placer côte à côte le récit et le registre. Le récit dit : je porte seul une charge dont personne ne veut. Le registre dit : une personne signe entre 89 et 100 % des fusions, refuse 2 à 3 fois moins que la moyenne de son milieu, et valide un volume de code que nul ne peut relire dans ces conditions. Ces deux énoncés ne

se contredisent pas — ils décrivent le même fait sous deux éclairages. Le premier le donne pour un sacrifice, le second pour une concentration de pouvoir sans contrôle. La différence entre les deux n'est pas psychologique, elle est politique : un sacrifice appelle de la gratitude, une concentration appelle des comptes.

L'aveu, du reste, ouvrait deux sorties, l'une et l'autre disponibles. Déléguer le pouvoir de fusionner, d'abord — la série montre qu'il s'est au contraire resserré d'année en année. Consulter avant de décider, ensuite : non pas recueillir des réactions après coup, ce que le projet fait abondamment et bruyamment, mais faire entrer dans la décision ceux qu'elle affecte *avant* qu'elle soit prise. C'est la conception participative, dont le chapitre « Penser contre soi » rappelle qu'elle est documentée depuis 40 ans et qu'elle a fait ses preuves avec des ouvriers qui ne programmaient pas. Aucune des deux sorties n'a été prise. Celle qui l'a été est la troisième, la seule qui ne coûte rien : fabriquer un récit légitimant.

Il faut citer ici l'étude qu'on m'opposera, et dire précisément ce qu'elle mesure. Agrawal et ses collègues ont étudié 661 projets libres et 171 projets d'entreprise pour savoir si les projets « à héros » — ceux où 20 % des contributeurs fournissent 80 % du travail — s'en sortent moins bien. Leur réponse est non. Ils constatent en outre que presque tous les projets le deviennent en grandissant, et concluent que les héros sont « bien plus répandus et bien plus précieux que la littérature ne le suggère », et qu'il faudrait savoir les retenir[13].

Quatre choses bornent cette conclusion, dont trois sortent de l'étude elle-même ou de sa confrontation au dossier. D'abord ce

qu'elle mesure : des délais — vitesse de fermeture des tickets et des demandes d'évolution. Elle ne mesure ni la maintenabilité, ni l'érosion de l'architecture, ni ce qu'il advient du projet quand le héros s'en va, c'est-à-dire précisément l'objet de ce chapitre, et ce que le facteur de bus cité plus haut mesure en sens inverse.

Ensuite, et c'est plus gênant pour l'objection : sur son propre indicateur, ce cas-ci échoue. Toute la portée rassurante de l'étude tient à ce que la présence de héros ne dégrade pas le traitement des demandes. Or le chapitre « Livrer, pas larguer » a mesuré ici l'inverse : les fermetures humaines d'*issues* ont fondu de 53 % entre 2020 et 2025 — non parce qu'il y aurait moins à faire : le rendement individuel des restants est constant, c'est l'effectif qui a fondu, et la couverture du flux entrant recule de 67 à 60 % — l'encours regonfle et le robot de péremption comble l'écart — pendant que 47 % des signalements, on l'a établi plus haut, ne se ferment jamais. Cette dégradation couvre exactement la période où la concentration s'accentue — d'une quinzaine de fusionneurs à deux. Un cas ne réfute pas une régularité établie sur 832 projets ; il en suspend l'usage : on ne peut pas invoquer pour défendre cette maison une étude dont la conclusion rassurante repose sur une variable qui, dans cette maison, s'est effondrée.

Ensuite, le résultat favorable de l'étude est confiné aux projets d'entreprise : dans les projets libres publics — la catégorie de celui-ci —, la présence de héros n'améliore rien. Or le héros d'entreprise est salarié, mandaté, remplaçable et comptable devant quelqu'un : c'est un héros encadré, et l'encadrement fait partie du résultat.

Enfin, ce n'est pas la concentration du travail que je conteste. Qu'une seule personne écrive ou valide l'essentiel du code n'a rien de scandaleux ; c'est même le cas courant, et l'étude vient de le rappeler. Ce que je conteste, c'est qu'elle n'ait de comptes à rendre à personne. Nul ne l'a désignée pour ce rôle, sa charge n'a pas de terme, et ses décisions ne peuvent être contestées devant aucune instance. C'est ce vide-là que le récit vient combler : il fournit une légitimité que personne n'a jamais accordée.

Car le récit du chef solitaire n'est pas un mensonge, et c'est ce qui le rend efficace. Un récit auquel son auteur ne croirait pas ne protégerait personne. Celui-ci est sincère, et il est même partiellement exact : la charge est réelle, l'ingratitude aussi. Ce qu'il accomplit, c'est un déplacement — il convertit un défaut d'organisation en qualité personnelle. Une fois la solitude devenue vertu, la partager devient une dégradation, et toute proposition de structure se lit comme une mise en cause. Le milieu tient ainsi son paradoxe : la même personne peut décrire sincèrement son isolement et refuser sincèrement les seuls remèdes qui le lèveraient. On n'en sort pas par la psychologie, on en sort par la structure — et l'on comprend au passage pourquoi la maintenance, qui est le contraire exact de l'exploit, ne trouve aucune place dans un imaginaire professionnel bâti sur la figure de l'homme seul capable de tout, imaginaire dont la sociologie des techniques a montré qu'il est aussi une identité de genre avant d'être une compétence[14].

Ce que serait l'ambition non héroïque tient alors en 4 engagements, dont aucun n'est spectaculaire. Le premier est d'inscrire la maintenance au devis. La norme ISO/IEC/IEEE 14764

classe les travaux de maintenance selon le moment — proactif ou réactif — et selon le but — corriger ou améliorer —, et donne à la maintenance préventive le rang d'une catégorie à part entière : corriger les fautes latentes avant qu'elles ne se manifestent en exploitation[15]. Une catégorie nommée est une catégorie qu'on peut budgéter, et qu'on peut reprocher à quelqu'un d'avoir omise.

Le deuxième est d'exiger le périmètre enclos, c'est-à-dire de n'accepter une fonctionnalité que si l'on peut énumérer ce qu'elle touche, et si cette liste est courte. La question n'est pas « est-ce que cela fonctionne ? » mais « que faudra-t-il ouvrir pour la réparer ? ». Une fonctionnalité dont la réponse est « à peu près tout » doit être refusée telle quelle, même bonne, même déjà écrite — surtout déjà écrite.

Le troisième est d'écrire la règle. L'intuition du développeur expérimenté ne se transmet pas par osmose, et le milieu recrute des gens de passage ; il faut donc que les critères d'acceptation soient écrits, publics et opposables — ce que le chapitre « Le pacte » proposera d'inscrire dans une institution. Ce n'est pas une préférence, c'est un résultat mesuré : l'enquête menée auprès des mainteneurs de 104 projets libres populaires devenus obsolètes a cherché ce qui sépare les projets qui survivent de ceux qui meurent, et deux pratiques ressortent, associées à la survie — l'existence de règles de contribution écrites, et l'intégration continue[16]. La structure ne garantit pas la vie d'un projet, mais son absence accompagne sa mort.

Le quatrième est d'imposer une culture et une éthique de travail, parce qu'elles se reproduisent d'elles-mêmes. Un code déjà

enchevêtré enseigne l'enchevêtrement : le contributeur qui arrive imite ce qu'il lit, et il lit du désordre. C'est la loi de Conway du chapitre « Le comité n'inventera rien » appliquée au temps plutôt qu'à l'espace — un projet reproduit sa propre structure passée. Rompre cela ne se fait pas par un rappel à l'ordre annuel : cela se fait en refusant une fois, publiquement, une contribution qui fonctionne.

Il faut nommer le prix de tout cela, car il est réel et il se paie immédiatement. Refuser une contribution offerte gratuitement, c'est décourager un bénévole. Exiger le périmètre enclos, c'est allonger le délai d'une fonctionnalité attendue. Budgéter le préventif, c'est financer un travail dont personne ne verra le produit — la définition même de ce que le chapitre « Compter les absents » appelait le bug évité, qui ne laisse aucune trace. L'ambition non héroïque paie en visible ce qu'elle achète en invisible. C'est pourquoi elle est rare, et c'est pourquoi il lui faut un mandat qui la protège : aucun individu ne tiendra seul, année après année, une position dont tout le monde constate le coût et personne le bénéfice.

Ce livre a défendu une définition de la conception : rapprocher une situation existante d'une situation préférée. Elle porte une contrainte de durée qu'on oublie d'énoncer. Rapprocher, puis laisser l'écart se rouvrir dès qu'on a le dos tourné, ce n'est pas avoir conçu : c'est avoir fait une démonstration. La seule ambition qui vaille pour un outil de travail est qu'il serve encore quand son auteur aura cessé de s'y intéresser. Il n'y a là rien d'héroïque, et c'est précisément ce qui la rend difficile à obtenir d'un milieu qui ne récompense que les annonces. « Non

héroïque » ne veut donc pas dire modeste, ni prudent, ni sans envergure : cela veut dire débarrassé du modèle du héros, qui est un modèle de genre avant d'être un modèle de travail. L'ambition dont un outil a besoin ne ressemble pas à un sauvetage. Elle ressemble à quelqu'un qui revient, tous les jours, là où il est déjà venu.

Cette partie du livre a décrit un collectif : ce qu'il livre, ce qu'il décide, ce qu'il coûte à ceux qui le tiennent, et ce qu'il devrait refuser. Reste à examiner le mot au nom duquel il refuse de s'organiser — la liberté —, et c'est l'objet de la partie suivante.

---

1. Lehman, M. M. (1980). « Programs, Life Cycles, and Laws of Software Evolution ». *Proceedings of the IEEE*, 68(9), p. 1060-1076 https://doi.org/10.1109/PROC.1980.11805 . Les 8 lois portent sur les systèmes dits de type E — ceux qui s'exécutent dans le monde réel et dont l'usage modifie le monde qu'ils décrivent, donc tout logiciel d'application. Les 3 mobilisées ici — changement continu, complexité croissante, qualité déclinante — sont celles que les réplications ultérieures valident le plus régulièrement ; les lois portant sur l'autorégulation et la stabilité organisationnelle sont plus discutées. ↩

2. Lientz, B. P. & Swanson, E. B. (1980). *Software Maintenance Management*. Reading, MA : Addison-Wesley — enquête sur 487 organisations, à l'origine de la typologie corrective / adaptative / perfective ; la maintenance perfective y dépasse la moitié de l'effort total. La fourchette de 60 à 80 % du coût

du cycle de vie consacrée à la maintenance est le chiffre retenu par l'IEEE Computer Society, corroboré depuis par Erlikh (2000) et par les travaux de Boehm. Les proportions sont restées remarquablement stables à travers les changements de technologie. ↩

3. Star, S. L. & Strauss, A. (1999). « Layers of Silence, Arenas of Voice : The Ecology of Visible and Invisible Work ». *Computer Supported Cooperative Work*, 8(1-2), p. 9-30 https://doi.org/10.1023/A:1008651105359 — le travail invisible n'est pas un travail qu'on ne voit pas par hasard : c'est une catégorie produite par les systèmes de description eux-mêmes, qui rendent certaines tâches comptables et d'autres non. Prolongé, pour les infrastructures, par Star, S. L. (1999). « The Ethnography of Infrastructure ». *American Behavioral Scientist*, 43(3), p. 377-391 https://doi.org/10.1177/00027649921955326 — une infrastructure devient visible au moment où elle tombe en panne. ↩

4. Archéologie du dépôt Darktable (branche master, HEAD 59ed384a1 du 31 juillet 2026), commits hors fusion touchant `src/`, auteurs dédupliqués (nom et courriel). Correctif = message contenant fix, bug, regression, crash, segfault, leak ou une référence d'*issue* ; nouveauté = add, new, introduce, implement, support, allow, enable ; le solde (refactorisation, nettoyage, build — 62 % des commits du cœur) n'est classé ni correctif ni nouveauté. Rustine : correctif dont tous les fichiers touchés avaient pour dernier modificateur l'auteur lui-même, moins de 90 jours auparavant — résultat identique

aux seuils de 30 et 180 jours ; réparer son propre code ancien compte comme réparation vraie. Chiffres : 252 développeurs — 121 sans aucun correctif (48 %), 7 rustineurs exclusifs, 124 vrais réparateurs (49 %) ; par palier : 1-2 commits, 146 développeurs dont 32 % de réparateurs ; 3-14, 59 et 56 % ; 15 et plus, 47 et 96 %. Volume : 15 599 commits, 3 289 correctifs déclarés (21 %), dont 1 758 réparations vraies (11 %) — 1 623 signées par le cœur (92 %). Limites : classification par message, cohérente avec les heuristiques des chapitres précédents ; l'attribution par fichier approxime « introduit par soi » ; « réparer » est mesuré au commit, non à l'événement de clôture du traqueur. ↩

5. Traqueur GitHub du projet, mesure sur les *issues* : médiane de clôture 1,3 jour et 9e décile 22 jours parmi celles qui se ferment, 47 % restant ouvertes ; à comparer aux 7,4 jours de médiane des fils du forum d'entraide. Le chapitre « Le comité n'inventera rien » donne le détail de la méthode et la part fermée par le robot de péremption. ↩

6. Pierre, A. (2023). « Darktable : crashing into the wall in slow-motion », 15 février 2023 — « 2022 is the first year that I find myself practically unable to identify the cause of most interface bugs, because the working logic has become very obfuscated and the code incomprehensible ». L'article donne aussi, cycle de version par cycle de version, le rapport entre *issues* fermées et propositions fusionnées : 12 %, 12 %, 17 %, 30 %, 46 %, puis 21 %, 15 % et 12 % — le sommet est au second semestre 2021, l'effondrement suit en 2022-2023. ↩

7. Avelino, G., Passos, L., Hora, A. & Valente, M. T. (2016). « A Novel Approach for Estimating Truck Factors ». *IEEE 24th International Conference on Program Comprehension (ICPC)* https://doi.org/10.1109/ICPC.2016.7503718 — le *facteur de bus* est le nombre minimal de développeurs dont le départ laisserait le projet incapable de continuer. Sur un corpus de 1 932 projets, 16 % ont vu partir la totalité de leurs développeurs-clés, et le développement n'a été repris par d'autres que dans 41 % de ces cas. ↩

8. ISO/IEC 25010, *Systems and software Quality Requirements and Evaluation (SQuaRE) — Product quality model* (révision 2023) https://www.iso.org/standard/78176.html. La maintenabilité y est définie comme le degré d'efficacité avec lequel un produit peut être modifié, et décomposée en modularité, réutilisabilité, analysabilité, modifiabilité et testabilité. ↩

9. MacCormack, A., Rusnak, J. & Baldwin, C. Y. (2006). « Exploring the Structure of Complex Software Designs : An Empirical Study of Open Source and Proprietary Code ». *Management Science*, 52(7), p. 1015-1030 https://doi.org/10.1287/mnsc.1060.0552 — le coût de propagation est la proportion moyenne des éléments d'un système affectés par la modification d'un élément tiré au hasard, calculée sur la matrice de structure de conception. La comparaison Linux / Mozilla et la chute du coût de propagation de Mozilla après refonte servent de contrôle. La lecture économique de la modularité comme valeur d'option est développée dans Baldwin, C. Y. & Clark, K. B. (2000).

Design Rules, Vol. 1 : The Power of Modularity. Cambridge, MA : MIT Press. ↩

10. Propositions de code du dépôt Darktable-org/Darktable, interrogation de l'interface de programmation GitHub (`type:pr is:merged` contre `type:pr is:unmerged is:closed`, par année de création ; 2016-2026 relevés le 24 juillet 2026, 2012-2015 le 1er août). **Correction des fausses clôtures** : une proposition peut être fermée sans fusion parce que le fusionneur a intégré son contenu à la main (rebasage ou reprise hors de la branche) ; le balayage des commentaires de clôture des 1 583 propositions fermées sans fusion (motifs « manually merged », « merged manually », « cherry-picked », « rebased and merged », etc.) en identifie 211, concentrées là où elles déforment le plus : 32 en 2018, 30 en 2019, 54 en 2020. Ces propositions sont requalifiées en acceptations ; les taux du corps du texte et du graphique sont les taux corrigés. Séries brutes, fusionnées / fermées sans fusion (fusions manuelles) : 92/30 (2) en 2012, 227/61 (2) en 2013, 318/67 (5) en 2014, 182/73 (9) en 2015 — l'ère plurielle refuse dans les ordres de grandeur de la littérature —, puis 238/55 (5) en 2016, 161/34 (2) en 2017, 246/88 (32) en 2018, 723/170 (30) en 2019, 1 635/231 (54) en 2020, 1 485/181 (14) en 2021, 1 163/133 (13) en 2022, 1 495/139 (18) en 2023, 1 083/106 (14) en 2024, 864/91 (7) en 2025, 821/97 (4) en 2026. Soit, après correction, 10 944 propositions acceptées contre 1 345 réellement refusées, 10,9 % de refus sur quinze ans. Deux réserves : le taux ne porte que sur les propositions tranchées, les propositions encore ouvertes étant exclues ; et la fermeture sans fusion ne distingue pas le refus motivé de

l'abandon par l'auteur ou de la fermeture automatique. Ces réserves jouent dans le même sens et ne changent pas l'ordre de grandeur de l'évolution ; la correction des fusions manuelles, qui joue en sens inverse, est faite. ↩

11. Gousios, G., Pinzger, M. & van Deursen, A. (2014). « An Exploratory Study of the Pull-Based Software Development Model ». *Proceedings of the 36th International Conference on Software Engineering (ICSE)*, p. 345-355 https://doi.org/10.1145/2568225.2568260 — sur le corpus GHTorrent, le taux de fusion moyen s'établit autour de 72 %, et environ 13 % des propositions sont rejetées pour un motif technique. Les bases ne sont pas strictement identiques à la mesure ci-dessus, qui exclut les propositions encore ouvertes ; la comparaison vaut comme ordre de grandeur, non comme égalité de méthode. ↩

12. Meindl, J. R., Ehrlich, S. B. & Dukerich, J. M. (1985). « The Romance of Leadership ». *Administrative Science Quarterly*, 30(1), p. 78-102 https://doi.org/10.2307/2392813 — les auteurs montrent que la performance d'une organisation est spontanément attribuée à ses dirigeants au détriment des autres facteurs (équipe, secteur, conjoncture), et analysent ce biais comme un raccourci de mise en sens : la causalité réelle est complexe, la causalité par un homme est immédiatement racontable. La notion a été reprise et prolongée depuis, notamment dans le numéro spécial *The Leadership Quarterly* consacré à sa mémoire (2011). ↩

13. Agrawal, A., Rahman, A., Krishna, R., Sobran, A. & Menzies, T. (2018). « We Don't Need Another Hero? The Impact of

"Heroes" on Software Development ». *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, p. 245-253 https://doi.org/10.1145/3183519.3183527 ; prépublication https://arxiv.org/abs/1710.09055. Les auteurs définissent un projet « à héros » par la règle des 80/20, étudient 661 projets libres et 171 projets d'entreprise, et concluent : « Software heroes are far more common and valuable than suggested by the literature, particularly for medium to large Enterprise developments. » Leurs variables de résultat sont des délais de traitement (fermeture des anomalies et des demandes d'évolution) ; ils relèvent eux-mêmes que l'effet favorable ne s'observe que sur les projets d'entreprise, la présence de héros n'améliorant pas le taux de traitement des demandes d'évolution dans les projets libres publics. ↩

14. Faulkner, W. (2007). « Nuts and Bolts and People : Gender-Troubled Engineering Identities ». *Social Studies of Science*, 37(3), p. 331-356 https://doi.org/10.1177/0306312706072175 — l'identité professionnelle de l'ingénieur se construit sur une valorisation du « purement technique » au détriment du relationnel et de l'organisationnel, partage lui-même genré, et qui rend peu lisible — voire peu honorable — le travail de coordination et d'entretien. ↩

15. ISO/IEC/IEEE 14764:2022, *Software engineering — Software life cycle processes — Maintenance* https://www.iso.org/standard/80710.html. Les catégories de travaux y sont définies en croisant le moment du changement (proactif ou réactif) et son but (correction ou

amélioration), ce qui donne les 4 types : correctif, adaptatif, perfectif et préventif — ce dernier désignant la correction de fautes latentes avant qu'elles ne se manifestent en exploitation. ↩

16. Coelho, J. & Valente, M. T. (2017). « Why Modern Open Source Projects Fail ». *Proceedings of the 11th Joint Meeting on Foundations of Software Engineering (ESEC/FSE)*, p. 186-196 https://doi.org/10.1145/3106237.3106246 — enquête auprès des mainteneurs de 104 projets GitHub populaires devenus obsolètes, aboutissant à 9 raisons d'échec réparties entre l'équipe (manque de temps, manque d'intérêt, conflits entre développeurs) et le projet lui-même (obsolescence, concurrence). Les auteurs relèvent que l'adoption de règles de contribution et de l'intégration continue présente une association importante avec l'échec ou le succès d'un projet. Association mesurée, non causalité démontrée. ↩

Partie V — La liberté et ses conditions

24. La liberté dans le vide

« Le code est libre, tu n'as qu'à le corriger toi-même. » Tout utilisateur de logiciel libre a reçu cette phrase un jour, en réponse à un défaut signalé. Elle est parfaitement exacte, et elle ne sert à rien d'autre qu'à intimider. L'utilisateur qui vient de la recevoir ne sait pas programmer, n'a pas 10 ans pour apprendre le C, et le sait ; celui qui la prononce le sait aussi. La phrase n'informe pas : elle clôt. Elle convertit une liberté en fin de non-recevoir. Cette conversion n'est pas un accident de forum — elle est logée au cœur de la doctrine du logiciel libre. La démonter n'est pas un exercice de style : tant qu'elle tient, le logiciel libre restera un club, au lieu du service public qu'il prétend être.

Le logiciel libre repose sur 4 libertés : exécuter le programme, l'étudier, le redistribuer, le modifier. Ce sont de belles libertés, je leur dois ce que j'ai construit, et ce livre n'existerait pas sans elles : rédigé sur Linux, dans Visual Studio Code, publié sur un site web réalisé avec Hugo, sur un serveur Linux. Mais leur forme, d'abord : toutes quatre disent ce que personne n'a le droit de vous *interdire*. Des portes ouvertes — rien de plus. Aucune ne dit un mot de ce qu'il faut payer pour franchir la porte : savoir lire du C, avoir du temps libre, une machine de développement, des années de pratique. Leur public, maintenant : 3 de ces libertés — étudier,

modifier, redistribuer du code — ne s'exercent qu'en sachant programmer, et moins de 6 % de la population mondiale sait écrire du code[1]. Il faut donc appeler ces libertés par leur nom : ce sont des **libertés d'ingénieur**. Une licence libre lève la propriété intellectuelle sur le code — opération dont les ingénieurs savent tirer parti, et ils sont à peu près seuls. Pour les 94 % restants, la porte est ouverte et la pièce inaccessible ; le chapitre « Compter les absents » a donné le reste des chiffres — la moitié du monde ne sait pas installer un logiciel. Une liberté qui ne profite qu'à ceux qui possèdent déjà la compétence n'est pas une émancipation : c'est un privilège présenté comme un droit.

Cette objection n'est pas de moi, et elle est plus vieille que le logiciel. Isaiah Berlin distinguait la liberté *négative* — l'absence d'entrave : personne ne me bloque — de la liberté *positive* — la capacité effective de faire[2]. Les 4 libertés sont purement négatives, et le vocabulaire moderne de cette critique appartient à Amartya Sen : ce qui compte n'est pas le droit formel mais la *capabilité*, ce qu'une personne est réellement en mesure de faire et d'être[3]. Proclamer que chacun est libre d'acheter du pain ne change rien pour qui n'a pas de quoi le payer ; proclamer que chacun est libre de corriger le code ne change rien pour qui ne sait pas programmer. Martha Nussbaum a nommé exactement ce qui manque : les *capabilités combinées*, la conjonction d'une aptitude intérieure et des conditions extérieures de son exercice[4]. Une licence fournit la permission ; elle ne fournit ni l'aptitude, ni le temps, ni l'outil utilisable. Le logiciel libre prêche une liberté définie hors de tout contexte et de toute structure — une liberté dans le vide.

Virginia Held avait posé le diagnostic un étage au-dessus du logiciel : les théories du contrat, écrivait-elle, prennent pour paradigme « les circonstances idéales d'un chef de famille adulte et indépendant » et ignorent tout le reste — que chacun commence sa vie entièrement dépendant d'autrui, et que l'indépendance qu'on se prête repose sur un réseau de relations qui la rend possible[5]. Une licence est une pensée du contrat, et elle en hérite l'angle mort par construction : que les quatre libertés servent des égaux imaginaires et laissent la dépendance réelle à la porte n'est pas un accident de rédaction — c'est ce que la forme contractuelle produit partout où on l'emploie.

Et l'utilisateur, dans cette transaction ? Il en est exclu par le texte même. Toute licence libre l'énonce en majuscules : le programme est fourni TEL QUEL, SANS AUCUNE GARANTIE, pas même la garantie implicite qu'il convient à un usage quelconque. L'utilisateur n'a d'autre droit que d'exécuter le code en l'état — ni droit au support, ni droit aux corrections, encore moins à voir l'outil épouser ses besoins. Si de l'aide lui est fournie, c'est à la discrétion des développeurs, et la gratitude éternelle est attendue en retour. Prudence juridique, soit. Mais posez cette clause à côté du discours qui accompagne les mêmes logiciels sur leur page web — qualité professionnelle, alternative crédible aux outils commerciaux, rejoignez-nous. On attire l'utilisateur sur le terrain politique et éthique, la liberté, la souveraineté des données ; et quand il revient signaler que l'outil le fait souffrir, on lui oppose le terrain économique : travail bénévole, pas d'argent, pas de responsabilité — tu n'avais qu'à le corriger toi-même. L'argumentaire d'entrée ne parlait que de liberté, et en creux,

d'émancipation ; nulle part il ne disait qu'en installant, on renonçait au droit de trouver l'outil mauvais. Le mot « professionnel » a été prononcé, et la promesse non tenue : il y a quelque part une incohérence trompeuse. La liberté est vendue en vitrine ; la non-garantie est en majuscules, mais cachée dans le code source.

D'où vient un angle mort pareil ? De la généalogie du mouvement, et elle se lit encore dans ses réflexes. Le logiciel libre est l'enfant de la culture hacker — individualiste par nature, compatible avec le capitalisme par procuration, via la technophilie et le solutionnisme technologique. Dans cette culture, *partager tel quel* — son code, ses bidouilles — a été confondu avec *prendre soin de* — des besoins, des limites et des difficultés des autres — et la confusion a été enrobée de récits quasi hagiographiques. Quand on ne sait faire que du code, le code est la solution à tout. Or au moment où le mouvement naît, dans les années 1980, la fraction de l'humanité qui possède un ordinateur personnel est infime ; la liberté du code est le produit de ce privilège-là, pensée par lui et pour lui. Les minorités privilégiées sont toujours aveugles à leurs privilèges — c'est même à cela qu'on les reconnaît. Le logiciel libre est une liberté pour ceux qui ont les moyens de se l'offrir ; et le fait que l'étiquette ne porte pas de symbole monétaire rend la chose vicieuse, car le prix existe — en compétence, en temps, en machines — et ne pas l'afficher permet de soutenir qu'il n'y en a pas.

Il faut s'arrêter sur cette confusion, car elle est le sujet du livre entier et le moment est venu de le dire franchement. Écouter

avant de décider, caractériser le besoin avant la solution, vérifier la réception au lieu de la présumer, maintenir au lieu de larguer, répondre de ce qu'on livre : les 4 premières parties ont décrit, méthode après méthode, ce que *prendre soin* veut dire dans un métier technique. Elles l'ont fait sans prononcer le mot, parce qu'il fallait d'abord établir les gestes ; la conclusion les rassemblera sous leur vrai nom. On verra alors que le logiciel libre n'a jamais manqué de générosité — il a manqué de tout ce qui, dans le fait de prendre soin, n'est pas du partage de produit fini.

La même généalogie explique que le mouvement n'ait jamais pensé le travail. Son cadre implicite est celui du marché libre individualiste des *freelance* et des consultants : chacun est libre de lancer son projet, de bifurquer (*forker*) celui des autres, et le darwinisme social fera le tri. Soit — mais qui paie le travail ? Le capitalisme ne donne de valeur au travail que si son produit se vend, et la valeur suit la rareté : un produit dématérialisé, téléchargeable gratuitement à l'infini, a une valeur marchande nulle par construction. Le mouvement a donc inventé une structure de production communiste par nature — mise en commun du travail et du produit — et l'a encastrée dans un cadre idéologique capitaliste qui la prive de tout moyen de financer son propre travail. Les stratégies de financement qu'on lui voit essayer depuis 40 ans sont toutes précaires parce qu'elles héritent de cette contradiction, et le chapitre « Le travailleur invisible » a montré qui en éponge le déficit : le travailleur invisible, en heures non payées. J'y reviendrai au dernier chapitre de cette partie, car cette contradiction a une issue — mais elle suppose de penser

enfin ce que le mouvement n'a jamais voulu penser : la structure sociale du travail. La philosophie des techniques avait pourtant averti. Les artefacts ont une politique, écrivait Langdon Winner : la façon dont une technologie naît et s'organise incorpore des rapports de pouvoir, qu'aucune neutralité affichée n'efface[6]. Andrew Feenberg en tire la conséquence que le logiciel libre a manquée : la démocratisation qui compte n'est pas celle du produit fini, mais celle du *processus de conception* et des conditions de travail — et c'est précisément là que le mouvement, ayant libéré la licence, n'a rien libéré du tout[7].

L'objection orthodoxe fonde tout le mouvement. La licence, répond-elle, n'a jamais promis autre chose : les 4 libertés protègent contre l'enclosure propriétaire, point ; exiger d'elles l'utilisabilité ou le bien-être, c'est confondre une liberté avec une prestation. Et il faut concéder l'essentiel : arracher le logiciel à la propriété exclusive fut un combat d'émancipation réel, le socle de tout ce qui a suivi — y compris de mon propre travail. Mais l'argument prouve trop. Si la liberté formelle suffisait, le droit de propriété abolirait la pauvreté et le droit de vote ferait la démocratie ; personne ne soutient cela, précisément parce que la distinction entre le droit et les moyens du droit est acquise partout ailleurs. L'hôpital public en est la démonstration familière : il n'est pas le « concurrent » de la clinique privée, il est d'une autre nature — un service public existe pour donner aux gens les moyens d'exercer leurs droits sans conditions de ressources, parce qu'une liberté reste théorique tant qu'on n'a pas les moyens d'en faire usage. C'est la définition d'un service public, et c'est mot pour mot ce que Sen dit de la liberté. La

réponse n'est donc pas d'abattre les 4 libertés — elles sont un plancher nécessaire — elle est de leur refuser le statut de plafond. Ouvrir la porte est indispensable ; cela ne dispense pas de bâtir l'escalier. Et l'escalier — la formation, la documentation, l'outil transparent du chapitre « L'outil transparent », les conditions de travail du chapitre « Le travailleur invisible » — relève de la conception et de l'organisation, pas du droit. C'est un travail, quelqu'un doit le faire, et la licence est muette sur qui. Trente ans ont suffi à libérer le code ; il reste à libérer celui qui s'en sert, et cela ne se fera par aucune clause.

La même illusion a une face interne, tournée vers ceux qui produisent. Au nom de la même valeur, les projets s'organisent en liberté sans structure : pas de hiérarchie, pas de procédure, pas de responsable désigné. Le chapitre « Le comité n'inventera rien » a donné le verdict de Freeman — l'absence de structure ne supprime pas le pouvoir, elle le rend invisible — et il faut maintenant le pousser jusqu'à sa conséquence politique, car la symétrie avec l'usager est exacte. La liberté du producteur, telle que le milieu la pratique, est elle aussi purement négative : libre de chefs, libre de procédures, libre d'obligations. Or ce dont un collectif a besoin pour se gouverner est une liberté positive, au sens du même Berlin : la capacité de *se lier* — se donner des règles, prendre des engagements, désigner des responsables et pouvoir les démettre. Un groupe qui ne peut pas se lier ne peut rien promettre, ni à ses membres ni à ses usagers ; il ne décide pas, il dérive — et le chapitre « Livrer, pas larguer » a mesuré ce que dérive veut dire.

Le pouvoir qui occupe ce vide a de surcroît une propriété que le pouvoir institué n'a jamais : il est irrévocable. Freeman l'écrivait au cordeau : le pouvoir des structures informelles « ne leur a pas été donné ; il ne peut pas leur être retiré ». Un responsable désigné peut être démis ; une élite informelle, non — nul mandat à retirer, nulle élection à perdre, nulle instance à saisir. Il n'y a pas même d'adresse où porter le grief : demandez qui décide, on vous répondra « la communauté », et le chapitre « Penser contre soi » a compté ce que « la communauté » veut dire en pratique — deux fusionneurs, un seul pour l'essentiel depuis 2019. La seule sanction restante est la défection : partir, ou bifurquer — précisément le geste que le milieu traite en trahison. Et il faut retourner la médaille pour voir l'autre face : un pouvoir sans mandat ne peut pas être révoqué, mais il ne peut pas non plus être *obligé*. La licence a déclaré que personne ne devait rien à personne ; l'absence de structure garantit que personne ne répond de rien. Refuser de choisir qui exerce le pouvoir, concluait Freeman, n'abolit pas le pouvoir : cela abdique seulement le droit d'exiger que ceux qui l'exercent en répondent[8]. Des droits concentrés sans les devoirs qui vont avec — l'histoire a des noms pour ce régime, et « horizontalité » n'en fait pas partie.

Freeman, du reste, ne s'était pas arrêtée au diagnostic, et c'est la moitié de son essai que le milieu ne cite jamais : elle se terminait par des « principes de structuration démocratique » — déléguer une autorité définie, pour des tâches définies, par des procédures démocratiques ; rendre les délégataires responsables devant ceux qui les ont choisis ; distribuer l'autorité aussi largement que raisonnable ; faire tourner les charges ; attribuer les tâches sur des

critères rationnels ; diffuser l'information à tous, « car l'information, c'est du pouvoir » ; garantir l'accès égal aux ressources[8]. Autrement dit : les conditions institutionnelles de la liberté collective, écrites dès 1972 par le mouvement qui avait essayé l'horizontalité avant nous et payé pour voir. Le logiciel libre a hérité du refus des chefs ; il n'a pas hérité du mode d'emploi.

La liberté dans le vide produit ainsi deux vides jumeaux. Côté utilisateur : un droit d'accès sans la capacité d'en user. Côté producteur : une horizontalité de façade sous laquelle des élites informelles décident sans mandat et sans comptes. Dans les deux cas, le mot *liberté* recouvre exactement ce qu'il faudrait regarder — une compétence qui manque, un pouvoir qui se cache — et dans les deux cas, ce qu'il recouvre est une affaire de conditions, pas de permissions : des capabilités à bâtir côté usage, des institutions à bâtir côté production. La licence est muette sur les deux.

Voilà le vice de fond : le mouvement a défini la liberté comme l'absence d'entrave, et a cru en avoir fini. Tout ce qui décide qu'une liberté s'exerce ou reste lettre morte — la capacité de l'usager, les conditions du travailleur, la structure du collectif — est resté hors du champ. Ce hors-champ n'est pas vide pour tout le monde : il est occupé par ceux qui avaient déjà la compétence, le temps et les machines — les 6 %, et parmi eux une fraction encore plus étroite. Qui sont-ils, au juste ? J'ai posé la question à mes propres utilisateurs, deux fois, chiffres à l'appui. La réponse occupe le chapitre suivant, et elle explique beaucoup de choses.

1. Estimation à partir des données de l'Union internationale des télécommunications sur les compétences numériques (voir chapitre « Compter les absents ») ; je l'ai documentée dans l'analyse de ma première enquête utilisateurs (« Who are the Darktable users in 2020 ? », eng.aurelienpierre.com, 2023, § « Ability to interact with the computer ») https://eng.aurelienpierre.com/2023/01/who-are-the-Darktable-users-in-2020/. ↩

2. Berlin, I. (1958). *Two Concepts of Liberty*. Oxford : Clarendon Press ; repris dans *Four Essays on Liberty*, Oxford University Press, 1969 https://philpapers.org/rec/BERFEO (trad. fr. « Deux conceptions de la liberté », in *Éloge de la liberté*, Calmann-Lévy, 1988). J'emprunte à Berlin sa taxinomie, pas sa préférence : lui défendait la liberté négative. ↩

3. Sen, A. (1979). « Equality of What ? ». Tanner Lecture, Stanford University https://tannerlectures.org/lectures/equality-of-what/ ; développé dans *Development as Freedom*, New York : Knopf, 1999 https://www.worldcat.org/oclc/41404591 (trad. fr. *Un nouveau modèle économique : développement, justice, liberté*, Odile Jacob, 2000). ↩

4. Nussbaum, M. C. (2011). *Creating Capabilities : The Human Development Approach*. Cambridge, MA : Belknap Press https://doi.org/10.4159/harvard.9780674061200 (trad. fr. *Capabilités : comment créer les conditions d'un monde plus juste*, Climats/Flammarion, 2012). ↩

5. Held, V. (2006). *The Ethics of Care : Personal, Political, and Global*. New York : Oxford University Press, p. 81 et 13-14 https://doi.org/10.1093/0195180992.001.0001 : « taking the ideal circumstances of an adult, independent head of a household as paradigmatic and ignoring all the rest » ; « Every person starts out as a child dependent on those providing us care » (notre traduction dans le corps du texte). ↩

6. Winner, L. (1980). « Do Artifacts Have Politics ? ». *Daedalus*, 109(1), 121-136 https://www.jstor.org/stable/20024652 — les artefacts techniques incorporent des rapports de pouvoir ; certaines technologies sont politiques par leur forme même. ↩

7. Feenberg, A. (1999). *Questioning Technology*. Londres : Routledge https://doi.org/10.4324/9780203022313 (trad. fr. *(Re)penser la technique : vers une technologie démocratique*, Paris : La Découverte, 2004) — la « rationalisation démocratique » : démocratiser une technique, c'est ouvrir son processus de conception, non seulement diffuser son produit. ↩

8. Freeman, J. (1972). « The Tyranny of Structurelessness ». *Berkeley Journal of Sociology*, 17, 151-164 https://www.jofreeman.com/joreen/tyranny.htm ; voir le chapitre « Le comité n'inventera rien » pour le diagnostic. Verbatims (notre traduction dans le corps du texte) : « Their power was not given to them ; it cannot be taken away » ; « If the movement continues deliberately to not select who

shall exercise power, it does not thereby abolish power. All it does is abdicate the right to demand that those who do exercise power and influence be responsible for it » ; « Information is power ». Les 7 principes ferment l'essai sous le titre « Principles of Democratic Structuring » ; le corps du texte les condense en suivant l'ordre et la formulation de l'original. ↩ ↩

25. Ce que l'entre-soi produit

Qui fait le logiciel libre ? Faute de recensement, j'ai mesuré ce que j'avais sous la main : mes propres utilisateurs, interrogés deux fois avec la méthode du chapitre « Compter les absents » — 1 101 réponses de 59 pays en 2020, 1 253 en 2022, données anonymisées et code de traitement publiés avec les articles, pour que chacun vérifie[1]. Mesurer les utilisateurs, c'est mesurer le vivier d'où sortent les contributeurs : parmi les répondants de 2022, 65 déclaraient contribuer au projet, et l'échantillon dira dans un instant à quoi ce vivier ressemble. La mesure dépasse le cas : tout projet à la composition semblable produira les mêmes angles morts, et c'est pour cela qu'on les compte. L'archétype qui en sort, tel que je l'ai publié : un homme d'environ 50 ans, entre 25 et 65, diplômé ingénieur ou équivalent. À lui seul, cet archétype couvre 55 % de la base d'utilisateurs. Un utilisateur sur deux sait programmer — 46 % en 2020, 55 % en 2022, le filtre se resserre —, soit 7 à 8 fois la proportion de la population générale ; la première filière d'études, à elle seule, est l'informatique et le génie électrique, 45 %. Les femmes : 2 % — 25 personnes sur 1 243, et deux parmi les 65 contributeurs. Le métier que cet outil sert est pourtant majoritairement féminin : environ la moitié des photographes professionnels américains sont des femmes, et 60 à 75 % des diplômés en photographie. Entre le métier et son outil libre, la part des femmes est divisée par 25. Ce n'est pas une impression, c'est une mesure — et j'en avais tiré la conclusion dans l'article : le logiciel libre ne discrimine ni sur l'argent, ni sur

la race, ni sur l'âge, ni sur la religion ; il discrimine sur la littératie informatique, et apparemment sur le genre.

D'où vient cette composition ? Du chapitre précédent : la liberté du logiciel libre a un prix sans symbole monétaire — de la compétence, du temps, des machines — et il faut regarder qui peut le payer. La réponse tient en une subvention double. La première est de classe : l'enquête de référence sur les développeurs du libre donnait 97,5 % d'hommes, majoritairement employés dans l'informatique, et pour premier motif de contribution la stimulation intellectuelle — le jeu[2]. Et le jeu n'est pas qu'un motif d'entrée : c'est un critère de gouvernance. Sur le salon du projet Darktable, un modérateur rapporte avoir proposé des critères d'acceptation et de refus objectifs des contributions — rejetés, entre autres motifs, parce qu'ils « nuiraient au plaisir des développeurs »[3]. Le chiffre n'a rien d'un artefact d'une seule enquête : les grands recensements du logiciel libre donnent tous la même masculinité écrasante — de l'ordre de 1 % de femmes dans les premières études, 3 % dans le sondage d'une grande forge de code, autour de 10 % au plus haut des estimations récentes[4]. Le contributeur type a un salaire d'ingénieur qui paie la machine, l'appareil photo haut de gamme et les soirées disponibles ; la photographie est pour lui un loisir sans obligation de résultat — on peut jeter les ratés, travailler par petits lots, faire une pause quand on veut — et sa tolérance aux défauts d'interface est au-dessus de la moyenne, puisque le terminal ne lui fait pas peur. La seconde subvention est de genre, et elle se mesure aussi : dans les pays européens, les femmes fournissent environ 260 minutes de travail domestique non payé par jour, les

hommes 140 ; en France, l'enquête Emploi du temps attribue aux femmes 64 à 72 % des heures domestiques[5]. La sociologie a un nom pour cela depuis 1989 : la *seconde journée*, et son corollaire, l'écart de loisir — le temps libre masculin est plus long et moins interrompu. Le soir libre qui finance le logiciel libre est donc subventionné deux fois : par le salaire, et par une femme qui fait les courses pendant que Monsieur joue à l'ordinateur. Et cette scène-là est horodatée. Le chapitre « Le comité n'inventera rien » a tracé la répartition horaire des commits du projet, en heure locale de chaque auteur : le travail se concentre entre 17 heures et minuit, et plus d'un commit sur trois est signé entre 17 et 22 heures — l'heure des courses, du dîner et du coucher des enfants. Le battement de cœur du dépôt est le négatif exact de l'emploi du temps domestique : ce que le registre du code enregistre à son insu, c'est l'heure à laquelle quelqu'un d'autre tient la maison. C'est une structure, et on ne peut pas comprendre ce que produit le logiciel libre sans la nommer.

Cette double subvention ne fait pas que financer le loisir : elle le retourne en mépris de classe. Le contributeur pratique la photographie sans en dépendre, et il le sait — « je suis bien content de ne pas avoir à gagner ma vie avec » est une phrase qu'on entend dans ces milieux, dite sans malaise. De là un renversement des valeurs : le professionnel, celui pour qui l'image est un gagne-pain sous contrainte de délai, de volume et de client, devient à ses yeux le pauvre type qui a besoin d'un logiciel qui marche, tandis que lui-même jouit du luxe de l'outil qui résiste, du terminal, de la configuration à rallonge — signes d'appartenance à la caste de ceux qui savent. Lightroom devient

le logiciel des moldus ; le libre, celui des vrais sorciers — et l'expulsion se prononce sans détour : sur le salon du projet, à qui décrivait la difficulté d'éduquer les utilisateurs perdus, un habitué répondit que « ces utilisateurs-là devraient s'en tenir aux produits Adobe, s'il vous plaît »[6]. Le paradoxe est amer : au nom de l'émancipation, le libre est en pratique *moins* accessible au non-informaticien que le logiciel commercial qu'il combat, dont l'interface, elle, a été conçue pour être vendue à des gens sans compétence technique. Et cette barrière-là est invisible depuis le milieu qui la dresse, pour une raison de pure géométrie sociale : ceux qu'elle arrête n'entrent jamais dans la pièce. Ils ne déposeront pas de rapport de bug, ne viendront pas au forum raconter pourquoi ils ont renoncé — pour eux, l'informatique est un guichet qu'on subit, pas un terrain de jeu — et leur absence ne fait aucun bruit. C'est le volet social du chapitre « Compter les absents ». L'idée que, si votre passe-temps est le vrai métier d'un autre, vous ne serez jamais vraiment bon à ce jeu ne les effleure pas. Le surplus de temps, chez Veblen, était déjà le marqueur d'une classe qui prouve son rang en s'adonnant à des occupations improductives et en méprisant le labeur nécessaire[7] ; le geste est intact, transposé au code. Et l'histoire des techniques offre un précédent troublant : Ruth Cowan a montré que les appareils ménagers du XX^e^ siècle, censés libérer les femmes du travail domestique, en ont au contraire redistribué et intensifié la charge[8]. La cause que j'y lis — ils avaient été conçus par des gens exemptés de ce travail, pour une idée du foyer qui n'était pas celle de qui l'habitait — est la même qu'ici : des concepteurs exemptés du besoin qu'ils prétendent servir produisaient déjà, il

y a un siècle, le même résultat. Le logiciel libre n'invente rien ; il rejoue.

Car une pièce ainsi composée ne produit pas n'importe quoi : elle produit ce qu'elle est. La sociologie des techniques a nommé le mécanisme : l'*I-methodology* — le concepteur qui, sans même y penser, se prend lui-même pour l'utilisateur[9]. Le cas d'école de la littérature est une équipe masculine persuadée de concevoir pour « tout le monde », et qui configure ce tout-le-monde à son image[10]. Je n'ai pas besoin d'emprunter le cas d'école : le mien est public. En juin 2026, sur le forum de la communauté, une énième discussion sur l'interface de Darktable — ouverte le 22, fermée le 26, 246 messages en 4 jours, le sort habituel de ces fils — a produit une confession en règle. Le responsable des versions du projet y écrit, en toutes lettres : « laissez-moi le dire : j'aime beaucoup l'interface telle qu'elle est » ; puis, deux messages plus loin : « les développeurs de Darktable sont aussi des photographes. Et je suppose qu'on peut dire : les développeurs de Darktable aiment l'interface telle qu'elle est » ; et lorsqu'un utilisateur propose « Darktable est créé pour les nerds, par les nerds », la correction tombe : « Non. Le projet a choisi pour de bonnes raisons : "Darktable est créé pour les photographes, par les photographes" »[11]. Tout y est, sans que rien n'ait été arraché : le goût personnel érigé en critère, l'équipe prise pour échantillon des utilisateurs, et l'I-methodology promue devise officielle — *par* les photographes, *pour* les photographes, c'est-à-dire par nous, pour nous. Un habitué du même forum l'avait dit plus crûment des années avant : le logiciel est fait parce que ses développeurs voulaient un logiciel qui leur convienne. Le chapitre

« Écouter sans obéir » a établi pourquoi le concepteur ne peut pas se prendre pour l'utilisateur ; voici ce qui arrive quand une communauté entière le fait avec fierté.

Une telle pièce ne se rouvre pas d'elle-même : l'homogénéité, une fois installée, se reproduit toute seule. Le chapitre « Compter les absents » en a donné le premier ressort, le biais du survivant : ceux qu'on a rebutés ne sont plus là pour dire pourquoi ils sont partis, si bien que la pièce n'entend, en boucle, que des gens qui lui ressemblent et la confortent. C'est une chambre d'écho. Le deuxième ressort est le plus simple : inclure demande un effort conscient, et une pièce qui a fait de son goût sa devise ne le fournira pas. Toute communauté mêlant photographie et logiciel libre penchera par défaut vers l'homme aisé, éduqué, anglophone et à l'aise avec l'informatique, et il faudra un travail délibéré et quotidien pour tendre la main à celles et ceux que ce filtre exclut, sous peine de nourrir exactement le biais du survivant. Une communauté qui, au lieu de fournir ce travail, célèbre sa clôture comme une identité, verrouille sa composition et, avec elle, la liste des besoins qu'elle est capable d'apercevoir.

À ces deux ressorts, mon cas d'étude en ajoute un troisième, plus destructeur que les 3 autres : l'économie du prestige y est indexée sur la production, jamais sur la connaissance. On y gagne ses galons en publiant du logiciel — un module amusant, un outil-jouet que les autres peuvent essayer le soir même — et il n'existe aucun galon pour ce qu'on *sait* et ce que ça améliore effectivement dans l'outil. Or l'essentiel de ce qui s'est annoncé sur ce forum en traitement d'image repose sur une science de la couleur fausse et trahit une incompréhension profonde de la

théorie sous-jacente — ce qui n'empêche ni le plaisir d'usage, ni les félicitations, ni la réputation qui va avec. Le forum applaudit ce qui tourne, pas ce qui est juste ; entre le vrai et l'amusant, la pièce a choisi, et elle ne sait pas qu'elle a choisi.

La sélection inverse suit mécaniquement, et elle a des noms. Celui qui essaie d'enseigner, sources primaires à l'appui, doit contredire un par un des informaticiens et des amateurs pleins d'opinions sur l'art et la couleur, sans rien de substantiel pour les soutenir ; et l'asymétrie de ce travail a son aphorisme, la loi de Brandolini : réfuter une sottise coûte un ordre de grandeur de plus que la produire[12]. Trois personnes qui faisaient précisément ce travail d'éducation ont quitté ce forum pour des raisons connexes : Elle Stone, qui travaillait sur les profils de couleurs de longue date, Troy Sobotka, dont le filmic de Blender est à l'origine du mien, et moi. Ma fatigue se date, une fois de plus. En janvier 2022 — 4 mois avant le point de rupture que le chapitre « Le cercle fermé » a daté au 7 mai — j'ouvre un fil pour réexpliquer le fond une énième fois, et l'ouverture du message dit déjà l'usure : les gens qui ignorent les conseils reviennent se plaindre que « pourquoi-diable-avez-vous-changé-ce-qui-me-donnait-des-résultats-plus-vite »[13]. Parce que j'étais le programmeur derrière les fonctions, je ne pouvais jamais commencer par expliquer comment s'en servir : il fallait d'abord défaire ce que les gens croyaient savoir, et qui n'était au mieux que des inférences sur la nature et le mode de fonctionnement des outils basées sur leur comportement visible depuis l'interface. Ces lieux ne favorisent pas la rigueur intellectuelle ; ils favorisent les gens qui ont du temps libre. Et un jour, le bilan se fait tout seul : ce forum ne m'a rien apporté — pas

une joie, pas une conversation instructive — seulement du temps perdu à essayer de sauver les utilisateurs d'eux-mêmes, et de la meute des prêcheurs qui, en essayant d'aider, propage de fausses informations qui font perdre le temps de tout le monde.

Reste à voir ce que cette pièce ne peut *pas* produire, et c'est ici que la composition devient un destin technique. Une salle d'ingénieurs quinquagénaires venus jouer sélectionne des valeurs implicites : le jeu technique plutôt que le soin, la nouveauté plutôt que la maintenance, l'excitant plutôt que l'utile. L'éthique du *care* a décrit le mécanisme exact : prendre soin commence par *remarquer qu'un besoin existe* — et c'est précisément la phase qui échoue dans une pièce d'où les porteurs du besoin sont absents ; Tronto appelle « irresponsabilité privilégiée » le luxe de ne pas avoir à le remarquer[14]. Cette éthique n'est pas convoquée ici en passant : c'est le cadre dans lequel la conclusion relira tout le livre, et l'on peut déjà noter ce qu'elle a d'embarrassant pour le milieu — sa première phase, la plus élémentaire de toutes, est justement celle qu'une pièce homogène ne peut pas franchir. Et les historiens des techniques ont montré que le culte de la nouveauté dévalue la maintenance par le même geste culturel qui dévalue le travail domestique — la comparaison n'est pas de moi, elle est dans leur texte : l'obsession de l'innovation « obscurcit tout le travail, y compris le travail ménager, que les femmes, de façon disproportionnée, accomplissent »[15]. Les priorités mesurées de Darktable en sont la démonstration chiffrée : les contrôleurs MIDI — possédés par 10 % des utilisateurs, utilisés par 2 % — ont valu une refonte complète du système d'entrées, parce que c'est un jouet d'ingénieur ; les tablettes graphiques —

possédées par 37 %, l'outil standard du métier — ont une prise en charge si défaillante que seuls 6 % s'en servent, et cela n'a jamais été une priorité. Le jouet de la pièce a battu l'outil du métier, dans les chiffres.

Et il ne faut pas croire ce dialogue de sourds récent, ni lié à une personne : il est structurel, et il a une pièce d'archive datée. En mars 2016 — 6 ans avant les conflits que ce livre a racontés —, un utilisateur ouvrait sur le forum de la communauté un fil intitulé « le monde a-t-il besoin d'un autre éditeur photo incomplet ? ». Sa demande était du pur niveau des besoins, au nom d'utilisateurs absents : améliorer la stabilité et la vitesse de l'existant *sans ajouter d'options*, rendre les logiciels accessibles aux gens normaux. Les réponses, relues aujourd'hui, forment un catalogue presque parfait des registres de ce chapitre. Le registre du jeu : « les gens écrivent ce qui gratte leur propre démangeaison » ; « nous ne sommes pas payés — dire à quelqu'un de travailler sur quelque chose de plus utile est déplacé » ; « je fais ce logiciel pour moi, je le donne parce que je le peux, pas pour que vous vous en serviez ». Le registre de la dette inversée : « c'est le contrat implicite du logiciel libre : vous recevez le travail des autres et vous remboursez avec votre temps » ; « les profiteurs sont un fardeau, pas un actif ». Le registre du congédiement : « des remarques fondées sur l'ignorance », « et vous, que proposez-vous de faire pour aider ? ». Et quand un développeur — car il y en eut un, et c'est le contre-exemple qui honore le fil — soutint que celui qui donne un logiciel doit au moins à ses usagers de ne pas leur nuire, on lui répondit en citant *le texte de la licence* : « IL N'Y A AUCUNE GARANTIE POUR CE PROGRAMME » — la

clause juridique brandie comme réponse à une question morale. Un autre développeur finit par nommer ce qui venait de se passer : la question d'origine — « comment mieux employer les développeurs pour répondre aux besoins des utilisateurs ordinaires ? » — avait été dévorée par le débat sur ce que les développeurs doivent ou ne doivent pas ; le fil fut fermé le lendemain par un modérateur, sur une plaisanterie, la question restée sans réponse[16]. Tout y est, à 6 ans de distance : les deux niveaux de discussion — le besoin contre la démangeaison —, la demande de soin perçue comme une agression, et une « communauté » qui, sommée de choisir entre servir des besoins et rester un cercle d'intérêts, choisit le cercle sans même percevoir qu'il y avait un choix.

Encore faut-il lire ce fil pour ce qu'il est : de la parole. Les deux camps y existaient — c'est tout à l'honneur du fil —, mais ils n'existaient qu'en discours, et c'est la décennie suivante qui les a départagés, par le seul juge qui compte : l'énergie réellement investie. Le camp du besoin n'a produit aucune structure ; le camp de la démangeaison a produit le statu quo, et le statu quo a une pente. Dix ans plus tard, les mêmes plaintes reviennent, presque mot pour mot — la stabilité, l'inflation d'options, l'inaccessibilité aux gens normaux ; les miennes s'y sont ajoutées, et le chapitre « Le cercle fermé » racontera ce qu'il en a coûté. L'empilement de fonctionnalités s'est poursuivi — le chapitre « L'éloge du simple » l'a compté. Et la composition de la base d'utilisateurs, mesurée en tête de ce chapitre, donne la direction du mouvement : entre 2020 et 2022, la part de programmeurs parmi les utilisateurs est passée de 46 à 55 % — le filtre ne s'est pas desserré, il s'est resserré. Voilà

donc le tableau complet : un état d'esprit révélé en 2016 dans un fil de discussion, puis vérifié en trajectoire sur 10 ans de faits. Faute de structure pour porter l'autre camp, la parole est restée de la parole, et la pente a fait le reste.

Et il faut dire qui applaudit, car l'offre ne persisterait pas sans la demande. Parcourez les fils d'annonce d'une nouvelle version : l'enthousiasme y est immédiat, massif — et indifférencié. *Toute* nouveauté est acclamée, module inédit, option supplémentaire, mode expérimental, sans que soit jamais posée la seule question qui compte dans un outil de métier : est-ce que cela rend mes photographies meilleures ? C'est le biais technophile à l'état pur — la nouveauté technique comme valeur en soi, héritée de la généalogie du chapitre « La liberté dans le vide » — et il fonctionne comme une économie de l'acclamation : le développeur qui livre une fonctionnalité excitante est applaudi le soir même ; celui qui répare un cache pendant 4 ans ne fait pas d'annonce. Le chapitre « Le code est un passif » avait noté que les utilisateurs jugent la santé d'un projet à son agitation ; voici le mécanisme au complet : une salle qui applaudit tout ce qui bouge dresse ses développeurs à faire bouger n'importe quoi. Les critères de validation du chapitre « Poser le problème » — la tâche, le temps, le résultat — n'entrent jamais dans cette boucle de récompense ; l'applaudissement, lui, est instantané. On obtient exactement ce qu'on récompense, et cette salle-là récompense le neuf, jamais le juste. Quant aux photographes professionnels, ils ne sont pas sur GitHub ni sur IRC, là où les décisions se prennent — et s'ils ne savent pas ce qu'est IRC, c'est précisément mon propos. Qui, alors, peut dire ce dont ils ont besoin pour être

productifs ? Personne. Mais qui veut le savoir ? Personne non plus : puisqu'ils n'utilisent pas nos outils, c'est bien la preuve qu'ils sont des presse-boutons — et la prophétie s'autoréalise, chapitre « Compter les absents », version sociale.

D'où la thèse de ce chapitre : **ce que le logiciel libre produit est tout ce qu'un tel environnement peut produire — rien, dans cet environnement, ne fournit une raison de produire autre chose.** Là où des forces contraires existent — des utilisateurs qui pèsent, des concepteurs différents du concepteur type —, l'I-methodology recule, et l'on en connaît des cas documentés[17]. Mais en leur absence, elle opère par défaut, et l'environnement livre le produit que sa composition sélectionne. C'est le jumeau sociologique de la loi de Conway du chapitre « Le comité n'inventera rien » : le logiciel copie la structure de communication de l'organisation qui le construit — il copie aussi sa composition sociale. Un outil de métier produit par une pièce sans le métier, une interface pour experts du terminal produite par des experts du terminal, une maintenance dévaluée par une culture qui n'a jamais eu à faire le ménage : rien de tout cela n'est un accident.

L'objection protège quelque chose de vrai : c'est leur temps libre, ils ne doivent rien à personne ; celui qui gratte sa démangeaison personnelle ne doit de comptes qu'à sa démangeaison. Pour le projet personnel, elle a entièrement raison — le dépôt de week-end n'a de comptes à rendre à quiconque, et l'éthique du hobby y est pleinement légitime. Mais deux choses changent la donne. La première est le statut : quand un logiciel devient l'outil de travail de centaines de milliers de gens, il change de catégorie — il

devient une infrastructure, et l'on ne juge pas une infrastructure à l'aune du loisir de ses fabricants ; le chapitre « La liberté dans le vide » a montré que c'est le projet lui-même qui réclame ce statut, en se vendant comme alternative professionnelle. La seconde est plus profonde : la démangeaison n'est pas un hasard. Gratter sa démangeaison est innocent ; ce qui ne l'est pas, c'est la composition de la pièce qui décide *quelles démangeaisons existent*. Quand la pièce est homogène, l'itch management de Raymond devient un filtre social : les besoins des absents ne démangent personne, donc ils n'existent pas. La réponse n'est pas d'exiger des bénévoles qu'ils travaillent davantage — c'est de cesser de célébrer la clôture de la pièce comme une devise, et de poser la question que le dernier chapitre de cette partie affrontera : comment on ouvre la porte.

Avant d'y arriver, il reste à décrire ce qui se passe quand les absents insistent malgré tout — quand un utilisateur hors profil vient dire que l'outil le fait souffrir, et ce que la pièce lui répond. C'est l'objet du chapitre suivant.

---

1. « Who are the Darktable users in 2020 ? » (n = 1 101, 59 pays) https://eng.aurelienpierre.com/2023/01/who-are-the-Darktable-users-in-2020/ et « Who are the Darktable users in 2022 ? » (n = 1 253) https://eng.aurelienpierre.com/2026/07/who-are-the-Darktable-users-in-2022/, eng.aurelienpierre.com — données anonymisées et code de traitement téléchargeables avec les articles. L'archétype et sa couverture de 55 % sont la conclusion publiée de l'enquête 2020 ; le 2,0 % de femmes

(25/1 243), les 54,6 % de programmeurs et les 44,9 % d'informatique/génie électrique sont mesurés en 2022. Baselines : US Census, Pew, ITU, StatCounter. Réserve d'échantillonnage assumée : 68 % des répondants 2022 recrutés via les canaux Darktable — l'échantillon le plus favorable possible au projet. ↩

2. Lakhani, K. R. & Wolf, R. G. (2005). « Why Hackers Do What They Do », in Feller, J. *et al.* (dir.), *Perspectives on Free and Open Source Software*. Cambridge, MA : MIT Press https://doi.org/10.7551/mitpress/5326.003.0005 — n = 684 : 97,5 % d'hommes, 58 % d'emplois en informatique, première motivation « intellectuellement stimulant » (44,9 %), 73 % décrivant des états de flow. ↩

3. Même journal, 18 janvier 2023, verbatim du modérateur : « I've also suggested having some gates up for what to accept and what to turn away, and that was shot down for a variety of reasons, including "impacting developer fun" » (notre traduction dans le corps du texte). Le rapporteur est ici témoin critique — c'est lui qui proposait les critères ; la formule « developer fun » est celle des développeurs qui les ont refusés. La discussion d'origine n'a pas été retrouvée — ni sur le dépôt GitHub, ni sur le forum, ni dans la période couverte par le journal du salon : le refus a eu lieu hors des archives consultables, et le témoignage du modérateur en reste la seule pièce. Le critère invoqué, lui, n'est pas isolé : la conférence des 10 ans du projet, en 2019, énonçait déjà pour unique règle de « n'empêcher personne de coder quand il s'amuse » — voir le chapitre « Le cercle fermé ». ↩

4. Ghosh, R. *et al.* (2002). *Free/Libre and Open Source Software : Survey and Study — FLOSS Final Report.* Université de Maastricht https://www.researchgate.net/publication/264799695_The_freelibre_and_open_source_software_survey_and_study-FLOSS_final_report — 1,1 % de femmes ; GitHub (2017), *Open Source Survey* https://opensourcesurvey.org/2017/ — 95 % d'hommes, 3 % de femmes ; Trinkenreich, B. *et al.* (2022). « Women's Participation in Open Source Software : A Survey of the Literature ». *ACM TOSEM*, 31(4) https://doi.org/10.1145/3510460, 1-37 — environ 9,8 %. ↩

5. Hochschild, A. R. & Machung, A. (1989). *The Second Shift.* New York : Viking (trad. fr. *Le deuxième travail*) https://search.worldcat.org/title/18588639. Chiffres actuels : OCDE (2025), synthèse sur 23 pays européens — 262 minutes par jour de travail non payé pour les femmes, 141 pour les hommes ; INSEE, enquête Emploi du temps 2009-2010 — 64 à 72 % des heures domestiques accomplies par les femmes selon le périmètre. L'écart se resserre surtout parce que les femmes en font moins, pas parce que les hommes en font plus. ↩

6. Salon Matrix public du projet (journal cité au chapitre « Le cercle fermé »), 16 mars 2022 : « such users should stick to Adobe products please » (notre traduction dans le corps du texte). ↩

7. Veblen, T. (1899). *The Theory of the Leisure Class.* New York : Macmillan https://www.gutenberg.org/ebooks/833 (trad. fr.

Théorie de la classe de loisir, Gallimard, 1970) — le temps de loisir comme preuve de statut, et le mépris affiché du travail productif comme marqueur de caste. Usage prudent : écho conceptuel, non preuve sur le logiciel libre. ↩

8. Cowan, R. S. (1983). *More Work for Mother : The Ironies of Household Technology from the Open Hearth to the Microwave*. New York : Basic Books https://search.worldcat.org/title/9619681 — les techniques domestiques, présentées comme libératrices, ont redistribué et alourdi le travail des femmes ; conçues hors de l'expérience de celles qui l'accomplissaient. ↩

9. Akrich, M. (1995). « User Representations : Practices, Methods and Sociology », in Rip, A., Misa, T. & Schot, J. (dir.), *Managing Technology in Society*. Londres : Pinter https://shs.hal.science/halshs-00081749 — origine du concept d'*I-methodology* : le concepteur se prend, sans réflexivité, pour le représentant de l'utilisateur. ↩

10. Oudshoorn, N., Rommes, E. & Stienstra, M. (2004). « Configuring the User as Everybody : Gender and Design Cultures in Information and Communication Technologies ». *Science, Technology, & Human Values*, 29(1) https://doi.org/10.1177/0162243903259190, 30-63. Les deux projets qu'étudie l'article échouent l'un et l'autre — il vaut comme démonstration *a contrario* : sans contre-forces, l'I-methodology gagne. ↩

11. Fil « Another dt UI discussion », https://discuss.pixls.us/t/58799, ouvert le 2026-06-22, fermé

le 2026-06-26 après 246 messages. Messages 228, 230 et 233 (2026-06-26), verbatim : « So, let me voice it : I do like a lot the way the UI is » ; « Darktable devs are also photographers. And I suppose we can say : The Darktable devs like the way the UI is » ; « No you can not. The project has chosen for good reasons : “Darktable is created for photographers, by photographers” ». La doxa communautaire, plus ancienne (t/27818 https://discuss.pixls.us/t/27818, 2021) : « It’s made because the developers wanted to make a piece of software that suits them. » Et la prémisse était dite en coulisses dès 2023 — salon Matrix du projet, 19 janvier 2023, le mainteneur de la documentation : « I’d assume most (all) devs are users so they would consider themselves to be a key audience » — l’I-methodology énoncée comme une évidence, 3 ans avant la confession publique. Les coulisses disent autre chose que la scène : sur le salon Matrix du projet (journal cité au chapitre « Le cercle fermé »), le mainteneur de la documentation écrivait dès 2023-2024 — « quelque chose est ajouté qui était une mauvaise idée… cela encombre l’interface, et au lieu de retirer la mauvaise idée on ajoute un autre changement » (20 mars 2023) ; « je ne supporte plus les disputes sans fin avec [le mainteneur], qui finissent souvent en pire ergonomie à force de vouloir contenter tout le monde » (23 mars 2023) ; et, le 13 janvier 2024, en toutes lettres : « current interface is just bad » (notre traduction pour les deux premières). Le « nous aimons l’interface telle qu’elle est » public n’a jamais été le verdict des coulisses. ↩

12. Aphorisme d’Alberto Brandolini, ingénieur logiciel, formulé le 11 janvier 2013 : « The bullshit asimmetry [sic] : the

amount of energy needed to refute bullshit is an order of magnitude bigger than to produce it » (notre traduction dans le corps du texte). Adage, non résultat scientifique — mais discuté comme tel dans la presse scientifique : Williamson, P. (2016). « Take the time and effort to correct misinformation ». *Nature*, 540, 171 https://doi.org/10.1038/540171a . ↩

13. Fil « Aurelien said : basecurve is bad », https://discuss.pixls.us/t/29055 , ouvert le 23 janvier 2022 (archivé : https://web.archive.org/web/20260723151115/https://discuss.pixls.us/t/29055). Verbatim du message d'ouverture : « people who keep ignoring them often complain that Darktable is bad, scene-referred sucks, and why-the-fuck-did-you-change-display-referred-that-got-me-faster-results » (notre traduction, atténuée d'un cran, dans le corps du texte). ↩

14. Tronto, J. C. (1993). *Moral Boundaries : A Political Argument for an Ethic of Care*. New York : Routledge https://doi.org/10.4324/9781003070672 (trad. fr. *Un monde vulnérable*, La Découverte, 2009) — les 4 phases du care, dont la première, *caring about*, remarquer le besoin ; et l'« irresponsabilité privilégiée ». L'origine du champ est Gilligan, C. (1982), *In a Different Voice*, Harvard University Press https://doi.org/10.4159/9780674037618 — citée ici pour l'histoire des idées, ses résultats empiriques étant contestés. ↩

15. Russell, A. L. & Vinsel, L. (2016). « Hail the Maintainers ». *Aeon*, 7 avril 2016 https://aeon.co/essays/innovation-is-overvalued-maintenance-often-matters-more : « The most unappreciated and undervalued forms of technological labour are also the most ordinary… feminist theorists have long argued that obsessions with technological novelty obscures all of the labour, including housework, that women, disproportionately, do. » Développé dans Vinsel, L. & Russell, A. L. (2020). *The Innovation Delusion*. New York : Currency https://www.penguinrandomhouse.com/books/576816/the-innovation-delusion-by-lee-vinsel-and-andrew-l-russell/ . ↩

16. « Does the world need another incomplete photo editor ? », https://discuss.pixls.us/t/975 , 6-7 mars 2016 (archivé : https://web.archive.org/web/20250907023642/https://discuss.pixls.us/t/975). Citations traduites par nous ; verbatims : « People write (and share) things that scratch their own itch » (n° 3) ; « it's inappropriate to tell them to work on something more useful » (n° 8) ; « I do Filmulator for me. I give it away because I can, not because I think you should use it » (n° 14) ; « Freeloaders are a burden and not an asset » et « that's the implicit contract that comes with free software » (n° 10, un développeur du cœur de Darktable) ; « most of which are based on ignorance » (n° 21) ; la clause de non-garantie de la GPL citée en réponse (n° 28) ; le recadrage final « how could we better utilize developers to more completely meet the needs of typical users ? » (n° 27). Le fil contient aussi une vraie réponse d'architecture (n° 4) et

deux développeurs défendant les utilisateurs — l'éthos a gagné le fil, il n'y a pas régné sans combat. ↩

17. 3 régimes de contre-forces documentés. Les utilisateurs qui pèsent avant la conception : le projet UTOPIA, où 5 ouvriers du livre, mandatés par leur syndicat, ont conçu sur maquettes un système de composition parti de leur métier — le chapitre « Penser contre soi » en donne le récit et les sources ; le système pilote n'a jamais été commercialisé, faute de capitaux chez le fabricant — c'est la méthode, le design participatif, qui a survécu. Les concepteurs partis d'un autre utilisateur qu'eux-mêmes : la gamme de cuisine OXO Good Grips (1989-1990), dessinée pour les mains arthritiques de Betsey Farber et devenue un succès de masse — cas d'école du design universel, entré dans la collection du MoMA (Smart Design, *Good Grips*, 1989) https://www.moma.org/collection/works/3758 ; et l'« effet bateau de trottoir » : l'abaissement de trottoir arraché à Berkeley par les militants handicapés (premier bateau officiel en 1972) sert aujourd'hui aux poussettes, aux livreurs et aux valises — concevoir pour le cas que la pièce ne voit pas améliore l'objet pour tous (Blackwell, A. G. (2017). « The Curb-Cut Effect ». *Stanford Social Innovation Review*, 15(1) https://ssir.org/articles/entry/the_curb_cut_effect). Enfin, les utilisateurs qui conçoivent eux-mêmes : von Hippel documente des domaines entiers où l'innovation vient des usagers — 22 % des chirurgiens d'un échantillon de cliniques universitaires modifiant leur instrumentation, l'équipement de kitesurf inventé par les pratiquants puis repris par les fabricants (von Hippel, E. (2005). *Democratizing Innovation.*

Cambridge, MA : MIT Press https://doi.org/10.7551/mitpress/2333.001.0001 , chap. 2 et 7). ↩

26. La violence à trois étages

Un utilisateur bute sur une manœuvre, s'y reprend, échoue, et conclut : « je n'y comprends rien, je suis nul en informatique. » La phrase paraît anodine, une modestie ordinaire. Elle est le produit fini d'une violence en 3 temps. Premier temps, si ancien qu'on ne le voit plus : on l'a *obligé* à passer par un ordinateur pour une tâche qu'il accomplissait très bien sans. Deuxième temps : on a conçu un outil qu'il ne pouvait pas utiliser — un échec de conception, au sens des chapitres qui précèdent. Troisième temps, le plus efficace : on a obtenu qu'il en porte la faute. L'échec qu'il met au compte de sa propre incapacité a pourtant deux causes, et aucune n'est lui : au plus près, un outil mal conçu ; en amont, l'obligation d'en passer par cet outil, qu'il n'avait jamais demandée. La première violence enrôle, la deuxième exclut, la troisième fait signer à l'exclu la reconnaissance de dette. C'est la troisième qui verrouille l'ensemble, parce qu'un exclu qui se croit responsable de son exclusion ne proteste pas — il s'excuse. Nommer ces 3 temps, c'est donner à quiconque les subit, ou les inflige sans le voir, de quoi briser la chaîne.

Le premier étage est la conscription, et on ne le voit plus : personne, ou presque, n'a *choisi* l'informatique. Elle est arrivée comme condition d'accès à des choses qu'on faisait déjà — déclarer ses impôts, inscrire ses enfants à l'école, écrire à une administration, développer ses photographies. Ces tâches s'accomplissaient, et souvent bien, avec du papier, un guichet, un

laboratoire ; en une génération, la voie manuelle a été fermée — non pas concurrencée : *supprimée* — et l'ordinateur est devenu un passage obligé. Quand la seule façon de payer ses impôts passe par Internet, l'ordinateur cesse d'être une possibilité : il devient une compétence exigée et un coût caché — une taxe d'accès à ses propres droits. Le chapitre « L'instrument et la machine » a décrit ce que cette conversion forcée a fait aux artistes : elle les a chassés du médium même qu'ils travaillaient — le papier, le négatif, la matière, remplacés par des données que seuls les ingénieurs savent toucher. Le même geste a été imposé à tout le monde, et il a été habillé du seul mot qui interdise d'en débattre : le *progrès*. On n'arrête pas le progrès, n'est-ce pas ? Forcer des gens qui ne comprennent pas les ordinateurs à en passer par eux pour des tâches qu'ils maîtrisaient est une forme d'oppression ; la déguiser en marche inévitable de l'histoire est une forme de violence psychologique, car elle transforme une décision — prise par quelqu'un, quelque part, pour des raisons discutables de coût et de contrôle — en fatalité naturelle, contre laquelle protester paraît aussi absurde que protester contre la pluie. Personne n'a voté pour la fermeture des guichets. On a seulement constaté, un matin, qu'ils étaient fermés.

Le deuxième étage est déjà connu de ce livre : c'est l'échec technique des conscrits les moins armés pour la bataille numérique, mesuré au chapitre « Compter les absents » (la moitié du monde ne sait pas installer un logiciel) et diagnostiqué au chapitre « La liberté dans le vide » (la liberté d'un droit qu'on n'a pas les moyens d'exercer). La recherche sur la fracture numérique a montré que le vrai fossé n'est plus l'accès mais la *compétence* :

une fois tout le monde connecté, les écarts d'aptitude demeurent, et ils se stratifient par le niveau d'éducation[1]. « C'est gratuit, tu n'as qu'à t'en servir » masque cet écart au lieu de le combler, exactement comme le chapitre « La liberté dans le vide » l'a montré du mot *liberté*. Et Donald Norman a établi le versant conception, cité au chapitre « Il n'y a pas de problème technique » : ce qu'on appelle « erreur humaine » est presque toujours une erreur de conception, une faute que le produit rejette sur l'utilisateur, à défaut d'avoir été pensé pour lui[2]. Le deuxième étage, donc, n'a rien de fatal : c'est du mauvais design, réparable par du bon.

Le troisième étage est l'opération morale, et elle a un nom ancien. William Ryan l'a appelée *blaming the victim* : devant un échec social, on étudie la victime, on lui trouve une déficience, et l'on oublie la structure qui l'a produite[3]. Ryan visait des progressistes bienveillants, persuadés d'aider — et la cible est exacte, car le milieu du logiciel libre est bienveillant, convaincu d'offrir la liberté, et pratique pourtant l'opération sans en avoir conscience. Sa forme quotidienne est la culture du *RTFM* — « lis le manuel » —, la présomption d'erreur d'utilisateur, le « tu n'as pas assez cherché » opposé à qui signale un obstacle. Le trait distinctif de cette forme, ce n'est pas la sévérité — c'est qu'elle se maintient même quand le signalement se révèle *juste*. Un utilisateur trouve le thème par défaut illisible ; un développeur du cœur lui répond que c'est « une décision de conception pour le meilleur confort de nos utilisateurs » et l'invite, s'il n'est pas content, à ne plus poster, « ou peut-être n'êtes-vous qu'un troll » — et la version suivante du logiciel livre un thème à plus fort contraste, donnant

raison au troll présumé. Un autre signale un défaut de style ; on présume l'erreur d'utilisateur et on lui demande de « changer d'attitude ou de partir » — le défaut était réel, et il fut corrigé[4]. Le blâme précède l'examen et lui survit : c'est le signe qu'on ne cherche pas à savoir qui a raison, mais à faire taire qui dérange. Norman, encore lui, a nommé le résultat côté utilisateur : à force de se voir imputer les défauts du produit, l'usager les intériorise, et Norman parle explicitement d'*impuissance enseignée — taught helplessness* —, le pendant, appris et infligé, de l'impuissance apprise des psychologues[5]. Le mot est juste : l'incapacité n'est pas innée, elle est enseignée.

Comment un enseignement pareil tient-il ? Par ce que Pierre Bourdieu a nommé la violence symbolique : le pouvoir d'imposer des significations comme légitimes en dissimulant les rapports de force qui les fondent[6]. La violence n'est pas l'exclusion ; elle est d'obtenir que l'exclu tienne son exclusion pour légitime. Bourdieu et Passeron en avaient décrit le modèle à l'école : les élèves d'origine populaire lisent leurs faibles chances statistiques comme une incapacité personnelle, et *s'éliminent d'eux-mêmes* — l'auto-élimination, qui épargne au système d'avoir à exclure, puisque la victime s'en charge[7]. Je fais l'hypothèse — aucune étude ne suit le même individu de l'école au logiciel — que la phrase « je suis nul en informatique » est le même geste, prolongé. L'école a appris à une partie de la population qu'elle n'était pas faite pour l'abstraction technique ; le logiciel récolte cette conviction toute formée et la confirme, à chaque manuel illisible, à chaque message d'erreur en langue d'ingénieur. La psychologie fournit la mécanique fine de l'intériorisation :

l'attribution interne, stable et globale — non « cette manœuvre a échoué » mais « je suis incapable, avec tous les ordinateurs, pour toujours »[8] — et la spirale de l'efficacité personnelle, où la faible confiance engendre l'évitement, qui engendre l'incompétence confirmée[9].

Il faut ici renverser la charge de la preuve. L'intuition commune traite la passivité devant la machine comme un défaut personnel : l'utilisateur timide manquerait de curiosité ou d'audace, et devrait se secouer. Cinquante ans de recherche sur l'impuissance apprise disent autre chose : la passivité n'est le défaut de personne, elle est le point de départ de tout le monde. Ce qui s'apprend, c'est le contrôle — la découverte que ses propres actions ont des effets — et cet apprentissage n'a lieu que si l'environnement répond de façon prévisible aux essais[10]. L'audace devant un outil n'est donc pas une qualité d'entrée : c'est un produit de l'outil. Un logiciel qui ne cède jamais lisiblement aux gestes du débutant, qui punit l'exploration par la perte de données ou l'erreur cryptique, ne laisse jamais le contrôle s'apprendre. « Je n'y touche pas, j'ai peur de tout casser » n'est donc pas un trait de caractère : c'est la réponse exacte, et prévisible, à un outil qui n'a jamais récompensé un essai. On retrouve la sécurité du chapitre « Mesurer sans se mentir » par l'autre bout : le survol qui détruit des données n'est pas seulement dangereux, il enseigne l'impuissance.

L'objection est celle de la responsabilité personnelle, et elle a sa dignité : tout le monde ne fera pas l'effort d'apprendre, certains ne veulent pas, et l'on ne peut pas tenir un outil pour responsable de la paresse de qui refuse de s'y mettre. Il faut lui concéder son

noyau : la compétence a un coût, l'apprentissage est réel, et le chapitre « L'éloge du simple » a défendu la difficulté légitime — celle du métier, que l'outil n'a pas à supprimer. Mais l'objection confond deux choses que ce livre sépare depuis le début : la difficulté du métier et la difficulté de l'outil. Apprendre la photographie est un travail ; déchiffrer un message d'erreur écrit pour un ingénieur n'en est pas un, c'est une taxe.

La frontière entre les deux n'est d'ailleurs pas une affaire d'opinion : elle a un critère, et il est historique. La photographie existait avant l'ordinateur. Exposer, doser le contraste, retenir une zone et en déboucher une autre, corriger une dominante, choisir son grain : tout cela s'apprenait déjà en chambre noire, et cette difficulté-là — le métier — ne peut pas se réduire sans amputer la capacité de celui qui travaille. Rien, en revanche, sous l'agrandisseur, n'exigeait de comprendre un format de fichier, l'ordre d'un pipeline, un profil d'affichage ou le jargon d'un message d'erreur : cette difficulté-là est née avec l'outil, et elle se réduit sans rien enlever à personne. Si la distinction est devenue si difficile à voir, c'est que le numérique a avalé le métier entier. Pour qui a commencé la photographie avec un logiciel — la majorité des pratiquants, désormais —, l'outil et le métier arrivent soudés : toute difficulté se présente comme une difficulté logicielle, et l'on ne sait plus ce qu'on est en droit d'exiger de l'outil ni ce qu'on doit encore au métier. Seuls ceux qui ont pratiqué avant l'informatique peuvent mesurer la frontière de mémoire : ils savent ce qui appartenait à la photographie avant qu'elle n'entre dans l'ordinateur, et ce que les outils y ont ajouté de leur propre chef. Le critère de ce livre est le leur : est du métier

ce qui existait déjà sous l'agrandisseur ; le reste appartient à l'outil, et doit se justifier. Et surtout, l'objection prend le filtre pour la nature. Le cockpit du chapitre « Écouter sans obéir » le réfutait déjà : l'aviateur « moyen » n'existait pas, et c'est le siège réglable, non la sélection des pilotes, qui a résolu le problème. La démonstration existe aussi pour la compétence informatique elle-même : une grande école d'informatique américaine, confrontée à 7 % de femmes dans ses promotions, a porté ce chiffre à 42 % en 5 ans — sans abaisser le niveau, en changeant la culture d'accueil et les prérequis implicites[11]. 7 % n'était pas la mesure d'une aptitude naturelle des femmes à l'informatique ; c'était la mesure d'un filtre, et le filtre s'est révélé démontable. Ce qu'on prenait pour la carte du talent était la carte de l'accueil.

J'ai d'ailleurs la preuve par ma propre pratique. J'ai donné, 3 années durant, des cours particuliers sur mon logiciel — des élèves adultes, souvent diplômés, qui avaient lu la documentation et payaient pour apprendre : la population la plus motivée et la mieux équipée qu'on puisse rêver. Et beaucoup, malgré tout cela, n'arrivaient pas à démarrer le flux de travail. Devant ce constat, il n'y a que deux conclusions possibles : ou bien la plupart des gens instruits et volontaires sont des imbéciles, ou bien le design est mauvais. La seconde est la seule sérieuse. Quand la partie la plus favorisée de vos utilisateurs bute, ce n'est plus l'utilisateur qui est en cause — c'est la démonstration, faite en conditions optimales, que le deuxième étage de la violence est bien un défaut de conception, et non une carence des gens.

La violence à 3 étages est donc l'exact opposé de tout ce que ce livre a défendu. Concevoir, c'est résoudre le problème d'un

humain, et prendre en charge la traduction entre son besoin et l'espace des solutions ; la violence à 3 étages fait l'inverse — elle rate la traduction, puis facture l'échec à l'humain. Et il faut oser le mot juste pour ce que le milieu inflige ici : celui qui a le plus besoin qu'on prenne soin de lui y est traité comme celui qui le mérite le moins. La conclusion n'aura rien à ajouter à ce constat ; elle en tirera seulement la déontologie qui manque. Cette violence a une conséquence qu'on n'a pas encore nommée : elle est confortable pour qui l'exerce. Tant que l'incapacité est du côté de l'utilisateur, il n'y a rien à réparer dans l'outil, rien à remettre en question dans la pièce du chapitre « Ce que l'entre-soi produit ». Le blâme de la victime n'est pas seulement une injustice ; c'est le mécanisme par lequel un entre-soi se dispense de se réformer — et c'est pourquoi il ne s'arrête pas aux utilisateurs. Retourné vers l'intérieur, contre celui qui, du dedans, conteste, il devient le moteur d'une boucle qui se referme sur elle-même. C'est l'objet du chapitre suivant.

---

1. Hargittai, E. (2002). « Second-Level Digital Divide : Differences in People's Online Skills ». *First Monday*, 7(4) https://doi.org/10.5210/fm.v7i4.942 ; van Deursen, A. & van Dijk, J. (2011). « Internet skills and the digital divide ». *New Media & Society*, 13(6), 893-911 https://doi.org/10.1177/1461444810386774 — le fossé s'est déplacé de l'accès vers la compétence, et se stratifie par l'éducation. ↩

2. Norman, D. A. (2013). *The Design of Everyday Things*, éd. révisée. New York : Basic Books, chap. 2

https://jnd.org/books/the-design-of-everyday-things-revised-and-expanded-edition/ — l'« erreur humaine » est le plus souvent une erreur de conception. ↩

3. Ryan, W. (1971). *Blaming the Victim*. New York : Pantheon https://search.worldcat.org/title/110095 — étudier la victime pour y trouver le défaut, et ignorer la structure ; l'auteur y visait explicitement les réformateurs bienveillants. ↩

4. Fil « v4.4 appears grayed-out », https://discuss.pixls.us/t/40694 : à un utilisateur déplorant le manque d'attention porté à l'expérience utilisateur, un développeur du cœur du projet répond (2024-10-20) « don't bother to post here, or maybe you are just a troll ! » (archivé : https://web.archive.org/web/20260714060151/https://discuss.pixls.us/t/40694) ; le contraste du thème sera relevé dans une version majeure ultérieure. Fil « Problems with setting up a style in 4.8 », https://discuss.pixls.us/t/44459/40 : à un utilisateur jugeant le logiciel inutilisable, la modération répond (2025-10-06) « Then use something else. We don't need this kind of attitude here » (archivé : https://web.archive.org/web/20260723093928/https://discuss.pixls.us/t/44459/40), puis archive le fil. Deux cas, cités par le rôle des intervenants ; verbatims vérifiés dans les instantanés. ↩

5. Norman, *op. cit.* — Norman reprend l'*impuissance apprise* et forge la *taught helplessness* : l'utilisateur, à force de se voir imputer les défauts du design, apprend à se croire incapable. ↩

6. Bourdieu, P. & Passeron, J.-C. (1970). *La Reproduction : éléments pour une théorie du système d'enseignement*. Paris : Minuit https://www.leseditionsdeminuit.fr/livre-La_Reproduction-1952-1-1-0-1.html — la violence symbolique : « tout pouvoir qui parvient à imposer des significations et à les imposer comme légitimes en dissimulant les rapports de force qui sont au fondement de sa force ». ↩

7. Bourdieu, P. & Passeron, J.-C. (1964). *Les Héritiers : les étudiants et la culture*. Paris : Minuit https://www.leseditionsdeminuit.fr/livre-Les_H%C3%A9ritiers-1950-1-1-0-1.html — l'*auto-élimination* : les dominés convertissent une probabilité objective faible en incapacité subjective et renoncent d'eux-mêmes. ↩

8. Abramson, L. Y., Seligman, M. E. P. & Teasdale, J. D. (1978). « Learned Helplessness in Humans : Critique and Reformulation ». *Journal of Abnormal Psychology*, 87(1), 49-74 https://doi.org/10.1037/0021-843x.87.1.49 — le style attributif interne, stable et global. ↩

9. Bandura, A. (1977). « Self-efficacy : Toward a Unifying Theory of Behavioral Change ». *Psychological Review*, 84(2), 191-215 https://doi.org/10.1037/0033-295X.84.2.191 ; appliqué à l'informatique : Compeau, D. R. & Higgins, C. A. (1995). « Computer Self-Efficacy ». *MIS Quarterly*, 19(2), 189-211 https://doi.org/10.2307/249688 — faible sentiment d'efficacité → évitement → incompétence confirmée. ↩

10. Maier, S. F. & Seligman, M. E. P. (2016). « Learned Helplessness at Fifty : Insights from Neuroscience ». *Psychological Review*, 123(4), 349-367 https://doi.org/10.1037/rev0000033 — le renversement à 50 ans : la passivité est la réponse par défaut ; c'est le contrôle qui s'apprend, à condition que l'environnement réponde aux actions. ↩

11. Margolis, J. & Fisher, A. (2002). *Unlocking the Clubhouse : Women in Computing*. Cambridge, MA : MIT Press https://mitpress.mit.edu/9780262632690/unlocking-the-clubhouse/ — Carnegie Mellon a porté la part de femmes dans ses admissions en informatique de 7 % à 42 % en 5 ans par une réforme de la culture et des prérequis implicites, sans modifier les exigences académiques. ↩

27. Le cercle fermé

Les deux chapitres précédents ont regardé la pièce et sa violence envers le dehors. Celui-ci regarde ce que la même machine fait au-dedans — à ses propres membres, et d'abord à celui qui conteste de l'intérieur. Sa thèse tient en une définition : un cercle fermé n'est pas un groupe qui est d'accord ; c'est un groupe dont le système immunitaire attaque l'information dérangeante, et celui qui l'apporte. Le désaccord n'y est pas réfuté, il est expulsé ; et l'expulsion est ensuite réécrite pour que le cercle n'ait rien à apprendre d'elle. Cette boucle se lit deux fois : d'abord dans la littérature, qui l'a mesurée sur les grands projets ; puis sur le cas que je connais du dedans, vérifiable pièce par pièce puisque tout en est public et archivé.

La boucle générale d'abord, en corrélation et mécanisme, pas en causalité simple. Premier arc : l'agressivité est endémique et normalisée dans les hauts lieux du logiciel libre — sur la liste de diffusion du noyau Linux, la référence culturelle du milieu, l'analyse de 1 500 courriels de revue de code a trouvé l'incivilité installée comme registre ordinaire[1]. Le style est celui du combat rituel : on ne discute pas une idée, on l'attaque, et tenir sous l'attaque fait partie du jeu — une culture de l'argument-comme-affrontement que la linguistique a décrite bien au-delà du logiciel[2].

Mais cette agressivité, déjà étudiée parce que facile à voir, a aussi son pendant invisible : en matière de violence systémique, l'agressivité est celle qu'on remarque immédiatement, celle qui est facile à nommer. Et il ne faudrait pas réduire tout l'écosystème du libre au seul cas du noyau Linux. Il existe une violence plus sourde, plus sournoise, plus hypocrite, qui ne dit pas un mot plus haut que l'autre et reste courtoise au moins sur la forme. Le projet que j'ai quitté pratiquait peu l'invective — il usait : des signalements laissés pourrir jusqu'à la clôture robotisée, des critiques congédiées sans examen, des plaignants renvoyés à leur incompétence supposée. Une toxicité polie, presque invisible quand on survole les fils en ne jugeant que la forme. Il faut s'arrêter ici un moment sur ce que violence veut dire. Daisy Letourneur en donne une définition générale : la violence, c'est retirer des options aux gens[3]. Je restreins le spectre : c'est retirer des options, de façon arbitraire, sans nécessité démontrée. Si l'on accepte que « la liberté des uns finit là où commence celle des autres », la violence est toute forme de restriction de la liberté de l'un bien avant qu'elle ne commence à empiéter sur la liberté de quiconque. Ainsi, empêcher les gens de polluer l'environnement partagé par tous n'est pas une violence envers les pollueurs, c'est simplement protéger le droit de tous à vivre dans un environnement qui ne rend pas malade.

Mon cas d'étude ne pratiquait presque pas l'agressivité ouverte — le seul membre du cœur au registre réellement abrasif avait quitté le cœur depuis des années, pour ne plus maintenir qu'une bibliothèque tierce. Sur le forum comme sur le dépôt, on est poli, et c'est presque pire. La violence y prend la forme jumelle, que la

recherche connaît tout aussi bien. L'incivilité de basse intensité, d'abord, définie précisément par l'ambiguïté de son intention — assez déniable pour rester insanctionnable[4]. L'ostracisme, ensuite — ignorer, écarter, laisser sans réponse — dont la psychologie a montré qu'il active les mêmes régions cérébrales que la douleur physique et atteint 4 besoins fondamentaux : l'appartenance, l'estime de soi, le contrôle, le sens de sa propre existence[5]. Puis la violence symbolique, dont le chapitre « La violence à trois étages » a décrit le fonctionnement sur le grand public : imposer une lecture de la situation qui fasse porter le défaut par celui qui le subit. Retournée vers l'intérieur, elle prend une fonction supplémentaire, et c'est celle qui intéresse ce chapitre. Quand un utilisateur signale une panne et qu'on lui répond qu'il s'y prend mal, ou qu'il n'est pas le public visé, on ne se contente pas de le rabaisser : on convertit une information en défaut de personne. Le signalement était entré comme un fait portant sur le logiciel ; il ressort comme un fait portant sur l'utilisateur. Rien n'a été appris, et le cercle a gagné le droit de ne rien changer.

Au-dessus encore, la violence épistémique, qui ne consiste plus à blâmer l'utilisateur pour son expérience mais à lui retirer le droit de la raconter. Le chapitre « Écouter sans obéir » avait délimité le seul domaine où l'utilisateur fait autorité sans partage — ses buts, son contexte, ses douleurs : ce qu'il a vécu, lui seul en dispose. Le réécrire à sa place — « ce n'est pas un bug, c'est un choix de conception », « personne d'autre ne se plaint », « vous n'avez pas compris ce que vous avez vu », « les utilisateurs satisfaits sont moins bruyants » — lui retire la seule compétence que ce livre lui

reconnaisse entièrement. Christiane Rochefort en avait donné la formule que je n'ai pas su améliorer : l'oppresseur n'entend pas ce que dit l'opprimé comme un langage, mais comme un bruit[6]. Requalifier une parole en bruit — bruit de fond, plainte, humeur, incompétence — est l'opération la plus économique qui soit, puisqu'elle dispense d'y répondre et n'oblige pas à la contredire. La philosophie l'a nommée : assigner au locuteur un déficit de crédibilité qui dispense d'évaluer ce qu'il dit est une injustice épistémique[7]. Ce point précis a été mesuré. L'analyse qualitative de 203 fils de signalement sur GitHub, dont 81 toxiques, montre que la toxicité y naît d'abord de perceptions divergentes sur la gravité et la priorité du défaut — c'est-à-dire à l'endroit exact où se décide qui a autorité pour dire ce qui compte —, et que ces échanges réduisent la probabilité d'une suite concrète, comme le rattachement du signalement à une proposition de code[8]. Le signalement n'est pas seulement mal reçu : il cesse de produire des effets. Et à l'échelle de projets entiers, la prévalence de la toxicité varie en sens inverse du taux de résolution des signalements — association mesurée sur 2 828 dépôts, pas une causalité établie[9]. On en verra plus loin la forme extrême, appliquée non plus à l'utilisateur, mais au contributeur qui conteste.

Enfin la technique du questionnement civil inépuisable, assez répandue en ligne pour avoir reçu un nom, le *sealioning* : exiger poliment, indéfiniment, preuves et justifications — souvent sur des points élémentaires ou déjà documentés — jusqu'à épuiser la patience de la cible, qui, en craquant, se met seule en tort[10]. C'est l'économie de la toxicité polie : celui qui use n'est jamais

sanctionnable, sa violence est passive, invisible ; celui qui s'use finit par l'être, surtout s'il craque, ce qui justifie a posteriori la façon dont il a été traité par le groupe. Le dissident n'y est pas battu, il est éconduit — mille fois, courtoisement, jusqu'à ce qu'il sorte. Et le tri que ce régime opère a une direction, que le chapitre « Ce que l'entre-soi produit » a documentée sur les galons : la différence y est punie, l'exigence y passe pour de l'élitisme, l'adhésion à la rigueur scientifique pour un manque de sensibilité artistique. Il existe un mot pour un ordre qui récompense la conformité au milieu de gamme et évacue à la fois l'excellence et l'écart — la médiocratie : « les médiocres ont pris le pouvoir », et son injonction emblématique pourrait servir de devise au régime : « rangez ces ouvrages compliqués, les livres comptables feront l'affaire »[11]. Une médiocrité organisée n'a pas besoin d'agresser : il lui suffit de congédier.

Et le congédiement a son instrument de choix, qu'il faut démonter ici parce que le cas de ce chapitre en montrera l'application officielle : la police du ton. Traiter la colère d'un locuteur comme un vice de forme dispense, précisément, d'instruire ses raisons — le déficit de crédibilité de Fricker, appliqué au témoignage en colère[12]. Le procédé a une histoire, et elle est instruite : c'est celui qui rangea la colère des femmes sous le diagnostic d'hystérie — pathologiser l'émotion pour n'avoir pas à entendre ce qu'elle disait[13]. Audre Lorde l'avait démonté en une phrase, dès 1981, en rapportant l'injonction qui lui était faite — « dis-moi ce que tu ressens, mais sans trop de dureté, sinon je ne peux pas t'entendre » — et en retournant la question : est-ce ma manière qui l'empêche d'entendre, ou la

menace d'un message qui changerait sa vie[14] ? La colère est chargée d'information, disait-elle. Un collectif qui la juge au lieu de la lire se prive exactement de l'information que la sensibilité sociale du chapitre « Le comité n'inventera rien » sert à capter, et la forme irréprochable y devient un tribunal : on tranche au ton ce qu'on refuse de trancher au fond — je suis resté courtois, pas vous, donc j'ai raison. C'est pourquoi la fausse politesse n'est pas le contraire de la grossièreté : c'est un refus d'entendre, en tenue de soirée. Lire l'état de l'autre pour l'entendre est le geste par lequel commence tout ce qui mérite le nom de soin ; le lire pour s'en servir contre lui est le geste exact par lequel il finit.

Deuxième arc : cette norme — ouverture ou politesse — est rationalisée par l'idéologie. La rhétorique de l'ouverture requalifie la plainte en censure et protège les « personnalités difficiles ». Mais l'expression elle-même est un piège, et il faut la démonter au lieu de la reprendre, car elle juge un comportement hors de son histoire et de son contexte. Un comportement est un résultat : sur un instantané, sans la chaîne d'interactions qui y mène, la défense ressemble à de l'agression et la résistance à de la provocation. La psychologie de l'attribution a décrit le biais — l'observateur impute au caractère ce que l'acteur doit à la situation — et la méta-analyse oblige à le dire exactement : l'asymétrie, quasi nulle en général, subsiste précisément pour les événements négatifs, c'est-à-dire là où l'on distribue des blâmes[15]. Le chapitre « Le conflit bien tempéré » nommera la dette de conflit ; celui qui finit par éclater après des années d'usure polie fournit à l'instantané exactement l'image attendue — et l'usure, elle, ne figure sur aucune photo. Il y a plus grave.

Dans le régime poli du premier arc, l'étiquette ne protège pas seulement des agresseurs réels : elle en fabrique. La sociologie de la déviance l'a établi il y a 60 ans — « la déviance n'est pas une qualité de l'acte commis, mais une conséquence de l'application, par les autres, de normes et de sanctions » ; « le déviant est celui auquel cette étiquette a été appliquée avec succès »[16]. Il faut donc un critère, et il doit pouvoir servir à quelqu'un qui n'a aucune opinion préalable sur les personnes en cause — sans quoi chacun rangera les autres dans la case choisie d'avance. Ce critère tient en deux questions, portant non sur les gens mais sur ce qu'ils disent : que vise l'énoncé, et à quelle aune mesure-t-il ce qu'il dit ?

La première question sépare la personne du travail. Attaquer quelqu'un pour ce qu'il est, l'humilier, ouvrir les hostilités sans historique : rien ne rachète ces conduites, pas même d'avoir techniquement raison. Ce partage n'est pas de mon invention : la psychologie du travail l'a opérationnalisé il y a près de 40 ans sous le nom de *critique destructrice* — une critique non spécifique, sans égards dans la forme, et qui impute la mauvaise performance à des causes internes, c'est-à-dire à ce qu'est la personne plutôt qu'à ce qu'elle a fait[17]. Le troisième arc en citera les effets mesurés en revue de code ; ce qui sert ici est sa définition, car elle donne exactement la ligne cherchée. Il existe des gens qu'une communauté doit exclure — le raciste, le sexiste, celui qui agresse sans provocation préalable —, et les écarter n'est pas de l'étiquetage : c'est une police légitime, qu'une communauté ne peut refuser sans abandonner ses membres.

La seconde question sépare deux critiques qu'on confond constamment, et c'est elle qui manquait. Une critique visant une décision se mesure à l'une ou l'autre de deux aunes. Ou bien elle prend le projet au mot — vous annoncez ceci, vos registres montrent cela : elle n'importe alors aucune exigence étrangère, elle demande des comptes sur des engagements déjà pris, et le projet ne peut l'écarter sans se dédire. Ou bien elle mesure le projet à une aune qu'il n'a jamais adoptée — « faites-en un Lightroom gratuit » : c'est alors une demande, non un manquement. Recevable comme demande, et tout aussi refusable comme demande, car 12 bénévoles travaillant entre 17 h et 22 h ne rattraperont pas une multinationale ; le dire n'est pas fermer le cercle, c'est tenir un périmètre — exercice dont le chapitre « Poser le problème » a fait une discipline.

De là 3 cas, et 3 traitements. L'attaque contre les personnes s'exclut. La demande hors périmètre se refuse, en argumentant le périmètre et non la personne qui la porte. La critique immanente s'instruit — et elle seule, refusée, referme le cercle. Le mérite de ce partage n'est pas sa finesse : c'est qu'il porte sur des énoncés et non sur des individus. Une même personne produira les trois selon les jours, et c'est en jugeant les gens plutôt que leurs phrases qu'on fabrique les cases.

Ce partage a pourtant une limite, qu'il faut poser aussi nettement que la règle — sans quoi la règle devient une immunité. Juger les énoncés plutôt que les personnes vaut à l'échelle d'un échange ; cela cesse de valoir à l'échelle d'une trajectoire. Un acte isolé ne dit rien de celui qui l'a commis ; une série d'actes tenue des années durant, sans qu'une erreur soit jamais reconnue ni une

leçon jamais tirée, finit par en dire quelque chose — et ce quelque chose est un constat, non une insulte. Passé ce point, refuser de nommer la personne ne protège plus l'équité : cela protège l'individu contre le bilan de ses propres actes, et prive le collectif de la seule information qui lui serait utile. Le chapitre « Le travailleur incompétent » donnera à ce bilan sa forme rigoureuse : la compétence se juge sur un solde d'actions au cours du temps, pas sur un geste ponctuel isolé de son historique. Nommer quelqu'un au terme d'une série documentée n'est donc pas l'attaquer, c'est faire la synthèse de pièces déjà versées. La ligne est double : sur un échange, on juge des phrases ; sur une décennie, on peut juger un bilan — à condition qu'il soit fait de pièces datées, et opposables à l'intéressé.

Reste que ce partage, en pratique, n'est presque jamais appliqué de façon symétrique — et c'est là qu'est le scandale, non dans les catégories. La même conduite change de nom selon la position de celui qui la tient. Le milieu en fournit le cas le mieux documenté, et c'est sa référence culturelle : sur la liste du noyau Linux, l'incivilité mesurée au premier arc a été tenue, des années durant, pour le style d'un homme plutôt que pour un problème — jusqu'à ce que l'intéressé s'en excuse publiquement en 2018, se retire quelque temps, et que le projet remplace son « code de conflit » par un code de conduite[18]. Ce qui avait couvert cette conduite si longtemps n'était pas une doctrine sur la civilité : c'était la place de celui qui la tenait. Le remède écrit ne suffit pas à lui seul : la première étude empirique des codes de conduite du libre relève que leur application reste inégale[19]. Symétriquement, le contestataire reçoit l'étiquette au premier écart, et souvent sans

écart. Dans mon cas d'étude, « personnalité difficile » est ainsi le nom donné à quiconque dévie de l'ethos et de l'habitus du milieu — l'exigeant, le professionnel, le théoricien : tous « difficiles », parce qu'ils forcent le milieu à s'interroger sur lui-même. Et le traitement appliqué aux étiquetés — l'éconduite, l'usure, le *sealioning* — finit par produire chez certains la conduite qui justifie l'étiquette rétroactivement quand ils craquent bruyamment, plutôt que se retirer en silence. Là encore, la mesure existe : les destinataires d'une critique destructrice déclarent traiter les désaccords ultérieurs par la résistance ou l'évitement, non plus par la collaboration ou le compromis[17]. La prophétie s'autoréalise, et le cercle tient sa preuve.

Cette asymétrie d'application n'est pas un accident de mœurs ; c'est un effet de position, et il faut la poser en termes généraux, car on la retrouvera à chaque étape du cas. Le piège est ici que la violence de l'*establishment* (que je nomme violence de droite, conservatrice) peut s'exprimer sans expression : simplement par le maintien du statu quo, dans des institutions où l'on nie qu'elles soient injustes et excluantes pour certains, et où l'on refuse même d'en explorer l'hypothèse comme une simple expérience de pensée. Gagner un combat sans combattre, simplement par verrouillage des protocoles de décision, en n'ayant rien d'autre à faire qu'attendre que l'opposition se fatigue puis s'épuise, est certainement le plus odieux des privilèges. À l'inverse, la violence de l'opprimé (que je nomme violence de gauche, progressiste) est un acte de résistance qui utilise les moyens disponibles, bruyants, désorganisés et parfois maladroits, mais surtout : visibles[6], et c'est là tout leur tort. Le piège intellectuel qui nous attend ici

consisterait à croire que, parce que ces deux violences s'opposent l'une à l'autre, elles soient symétriques : elles ne le sont ni par les moyens, ni par les forces en présence. Le piège moral serait de condamner la violence visible pour sa forme, en ignorant ce à quoi elle s'oppose : un colosse inerte qui n'a pas la moindre envie de bouger. L'histoire contemporaine s'entend à nommer « résistants » ces gens que la France de Vichy et l'occupant allemand nommaient « terroristes », car ils assassinaient, sabotaient et portaient des armes en temps de paix (l'armistice ayant été signé le 22 juin 1940, la France était administrativement en paix). Il n'y a donc rien de plus bourgeois que de condamner « toutes les formes de violence » pour leur simple forme, sans regarder ni leur contexte, ni les autres possibilités réalistes de lutte offertes à leurs auteurs, car on sait parfaitement requalifier la violence a posteriori, pour la rendre tolérable quand elle nous arrange, aussi bien que distribuer des médailles à tous ces « terroristes ».

Reste à dire pourquoi le bilan des trajectoires n'est jamais fait, et la réponse tient dans une doctrine, énoncée publiquement. En juin 2019, à la rencontre annuelle du graphisme libre, les deux principaux artisans du projet donnaient une conférence pour ses 10 ans, consacrée pour partie aux « différentes approches de l'organisation d'équipe » ; c'est là qu'une diapositive résume en un mot la constitution du projet : *do-ocracy*. La même diapositive en donne l'unique article, et il faut le lire lentement : « s'il y a une règle, c'est de n'empêcher personne de coder quand il s'amuse »[20]. Le mot n'est pas une boutade. Il désigne un modèle de gouvernance répertorié, où ceux qui font sont ceux qui

décident, et où l'autorité se mesure au temps et à l'énergie investis[21].

Il faut voir ce que cette formule règle, et ce qu'elle interdit. Elle règle la légitimité sans élection ni mandat : faire, c'est être fondé à décider. Elle interdit, du même coup, toute évaluation de ce qui est fait — car si le titre à décider procède de l'acte, l'acte ne peut plus être jugé sans que le titre s'effondre. Critiquer le travail de celui qui fait revient alors à contester son droit d'être là. La *do-ocracy* ne distingue pas la quantité de la qualité, et ne le peut pas : elle n'a pas d'autre unité de compte que le fait d'avoir agi. D'où la réponse opposée à toute critique — « écris du code », « fais-le toi-même » — qui n'est pas une grossièreté, mais l'application exacte de la doctrine.

L'article, lui, dit plus encore que le mot, et sa forme mérite qu'on s'y arrête. La seule règle que le projet se donne est une *interdiction de refuser*. Elle ne fixe aucun critère d'acceptation, aucun seuil de qualité, aucune obligation d'entretien : elle borne l'autorité dans un seul sens, celui du non. Son critère, ensuite, est l'amusement de celui qui code — non le besoin de l'utilisateur, non la santé du logiciel, non la charge de maintenance laissée à ceux qui resteront, mais le plaisir du contributeur, posé comme ce que l'organisation doit protéger en premier. Le reste en découle avec une régularité presque mécanique. Le chapitre « L'ambition non héroïque » a mesuré un taux de refus des propositions qui s'effondre à mesure que le droit de fusionner se concentre : c'est la constitution appliquée — on ne refuse pas, puisque la seule règle écrite dit de ne pas empêcher. Et le chapitre « Ce que l'entre-soi produit » a recueilli, 4 ans plus tard, ce que devient cette règle

le jour où quelqu'un propose enfin des critères d'acceptation : ils sont écartés, entre autres motifs, parce qu'ils « nuiraient au plaisir des développeurs ». La diapositive de 2019 et le témoignage de 2023 disent la même chose à 4 ans de distance, l'un depuis la tribune, l'autre depuis le forum : le plaisir est le critère, et il prime.

C'est là l'origine du reste. Un bilan suppose une aune extérieure à l'acte, et la doctrine n'en admet aucune. Les instruments ordinaires de la responsabilité — l'élection, le jury, le mandat à terme — sont introuvables dans ces gouvernances informelles[22], et le chapitre « Le travailleur incompétent » a dit ce qui reste quand ils manquent : non pas l'indulgence, mais l'ostracisme, qui écarte sans jamais avoir à motiver. La *do-ocracy* ne produit donc pas une communauté sans pouvoir : elle produit une communauté où le pouvoir ne se discute pas, parce qu'il se confond avec le travail. Le chapitre « L'ambition non héroïque » en a mesuré la conséquence la plus nette : le droit de refuser les propositions s'éteint à mesure qu'il se concentre.

Il faut aussitôt rendre au projet ce qui lui revient, car la diapositive suivante complique le tableau — et la complication est plus instructive que le grief. Elle s'intitule « responsabilité », et elle énonce : « qu'on le veuille ou non, des gens dépendent de notre logiciel pour faire leur travail »[23]. La phrase est juste, et peu de projets libres la formulent aussi nettement. Il n'y a donc aucun déni : la dépendance des usagers est vue, nommée, assumée.

Reste à voir en quoi cette responsabilité est convertie. La diapositive lui donne un contenu unique : la promesse que les retouches anciennes resteront lisibles indéfiniment — vous retrouverez dans 10 ans le fichier annexe de votre photographie, et le logiciel en reproduira exactement le résultat. L'engagement est réel, techniquement coûteux, et le projet s'y est tenu. Mais observez le registre. Sur tout le reste de ce dont un usager dépend pour travailler — comprendre l'interface, obtenir une réponse à un signalement, ne pas voir son savoir-faire annulé à chaque version —, rien. La responsabilité est reconnue, puis acquittée dans le seul registre où elle s'acquitte en écrivant du code. La responsabilité vis-à-vis de l'utilisateur porte seulement sur les artefacts techniques que le logiciel a produits.

C'est le geste que le chapitre « Il n'y a pas de problème technique » décrit depuis le début de ce livre, appliqué cette fois à un devoir et non à un problème : une obligation envers des personnes reçoit une lecture technique, et devient une propriété de fichiers. Le choix n'a rien d'arbitraire — c'est celui qui laisse la constitution intacte. Tenir une promesse de format ne demande ni de consulter quiconque, ni d'évaluer un travail, ni de refuser une contribution : cela se fait en codant, donc sans jamais empêcher personne de coder. La responsabilité ainsi définie ne coûte aucune gouvernance. L'autre moitié — celle qui obligerait à arbitrer, à refuser et à entretenir — reste sans titulaire.

La conférence ne s'arrête pourtant pas là, et la diapositive suivante est la plus importante des trois. Elle s'intitule « leçons apprises », et elle pose au bilan la question du modèle : qu'a-t-on gagné au *laissez-faire* — le mot est d'eux ? Quatre réponses. Les 3

premières sont des gains réels : un développement rapide, des fonctionnalités auxquelles l'orateur n'aurait jamais pensé, des traductions dans les langues dont il ignorait l'existence. La quatrième tient en 2 mots et un émoticône : « de la dette technique :/ »[24].

Il faut mesurer ce que cette ligne établit, car elle dispense ce livre d'une bonne part de son travail. Le lien entre l'absence de refus et la dette technique n'est pas une thèse que je plaide contre le projet : c'est un constat que le projet a dressé lui-même, en public, devant ses pairs, et inscrit au 4e rang de son propre bilan en 2019, dans la conférence qui célébrait (avec une peu d'avance) ses 10 ans d'existence. Il n'y a rien à imputer : le diagnostic est signé.

Reste ce qui a suivi, et c'est là que le cas devient probant. Un diagnostic n'engage à rien tant qu'il ne modifie pas la règle qui l'a produit — et cette règle était l'unique article de la constitution. L'amender aurait voulu dire empêcher, parfois, quelqu'un de coder. Le chapitre « L'ambition non héroïque » donne la suite en chiffres : le taux de refus des propositions de code, qui était de 19 % l'année de cette conférence, tombe à 10,9 % en 2021, puis à 8,5 % en 2023. Après le constat, le laissez-faire n'a pas reculé : il s'est accentué. La leçon a été apprise au sens strict du mot — énoncée, écrite, présentée devant l'assemblée du logiciel graphique libre — et jamais appliquée.

On reconnaît là une figure que ce livre a déjà croisée deux fois. Le chapitre « Le culte du cargo » décrit des pratiques dont on reproduit la forme après en avoir perdu la raison ; le chapitre « Poser le problème » suit la méthode agile le long de ce trajet,

jusqu'à ce que ses propres signataires la désavouent. Le laissez-faire est la version gouvernementale du même mouvement. La règle a réellement fonctionné — dans les conditions de ses débuts : peu de code, peu d'usagers, aucun héritage à tenir, et l'erreur sans conséquence. Ces conditions ont disparu ; la règle est restée. Ce que la loi de Lehman, citée au chapitre « L'ambition non héroïque », énonce du logiciel — il change sans cesse, donc il faut retravailler ce qui le tient — vaut aussi de sa gouvernance : une constitution écrite pour quelques personnes qui s'amusent ne régit pas un outil dont des professionnels dépendent.

La comparaison a toutefois une limite, et elle joue contre le cas plutôt qu'en sa faveur. Le culte du cargo, chez Feynman, suppose l'ignorance : on répète le geste sans savoir ce qui lui donnait son efficacité. Ici, personne n'ignorait rien — la 4e ligne du bilan nommait le coût, et la 3e diapositive nommait la dépendance des usagers. Il ne s'agit donc pas d'un rite maintenu faute d'avoir compris, mais d'un rite maintenu bien qu'on eût compris. Le chapitre « Penser contre soi » montrera que ce milieu sait, parfois, se regarder en face ; le cas ajoute ici la précision qui manquait : se regarder ne sert à rien tant que rien n'oblige à en tirer les conséquences.

Il manque un trait pour comprendre pourquoi tout, dans un tel milieu, dégénère en combat et doit en passer par là pour peu qu'on se soucie de la direction que prend le projet : il n'existe aucun tribunal. Les structures « sans pouvoir » du chapitre « Le comité n'inventera rien » n'ont pas seulement des élites informelles ; elles n'ont pas non plus d'instance d'arbitrage — pas de juge, pas de procédure d'appel, personne d'habilité à trancher

un désaccord sur l'orientation du projet, un choix de conception ou une implémentation, et surtout personne pour congédier l'incompétent récidiviste, ni d'ailleurs pour mesurer l'incompétence. Or les désaccords, eux, existent, et le chapitre « Le mot et la chose » a montré qu'ils ne s'éteignent pas d'eux-mêmes. Faute d'arbitre, chacun se bat donc avec les armes dont il dispose, pour ce qu'il croit juste : l'un a les droits de fusion, et fusionne ; l'autre a la modération du forum, et ferme les fils ; un troisième a du temps libre en abondance, et gagne à l'usure ; le dernier n'a que la parole publique, et il ne lui reste qu'à faire campagne. C'est un far-west, mais par absence d'institution judiciaire : le conflit s'y règle par les moyens, jamais sur le fond, et la victoire revient au plus disponible ou au mieux doté en droits d'accès, indépendamment de la validité ou de la rationalité des positions. On retrouve là l'asymétrie des deux violences décrite plus haut : qui tient les droits d'accès n'a rien à faire pour l'emporter, il lui suffit de ne pas bouger ; qui n'a que la parole doit agir, donc se montrer, donc s'exposer au reproche de la forme. Le chapitre « Décider, puis tenir » donnait aux décisions une règle de clôture ; ici, la seule règle de clôture est l'épuisement d'un des camps. On reconnaîtra chacune de ces armes dans le cas qui ferme ce chapitre.

Troisième arc : les coûts de la norme ne sont pas également répartis. L'étude des revues de code destructrices montre qu'elles démotivent tout le monde, mais qu'elles poussent vers la sortie les femmes et les personnes non binaires dans une proportion nettement supérieure[25] ; et l'analyse de millions de contributions montre que les contributions des femmes sont mieux acceptées

que celles des hommes — tant que leur genre n'est pas identifiable, l'acceptation chutant sous celle des hommes dès qu'il l'est[26]. C'est aussi la réponse à l'objection de la franchise — le milieu serait simplement direct, et les susceptibles n'ont qu'à s'endurcir : une « franchise » dont les coûts mesurés tombent asymétriquement sur certains groupes n'est pas un style de communication, c'est un filtre de population.

Quatrième arc, qui referme la boucle : ceux qui partent ressemblent précisément à ceux qui manquaient déjà (chapitre « Ce que l'entre-soi produit »), le tri se renforce, et la norme qui a fait fuir se trouve confirmée comme « la culture du milieu ». Une norme née dans un entre-soi, rationalisée par ses valeurs, dont les coûts expulsent qui ne lui ressemble pas : le cercle se referme tout seul, sans qu'aucun membre n'ait besoin de le vouloir.

Il faut aussitôt compléter le tableau, car la violence du cercle ne descend pas que vers les minorités : elle remonte aussi. L'étude systématique des fils toxiques sur GitHub a établi que le motif dominant n'y est pas l'affrontement entre développeurs, mais **l'utilisateur en position d'exigence contre le mainteneur**[27] — le siège décrit au chapitre « Le travailleur invisible », mesuré. Dans un cercle fermé, chacun est tour à tour tireur et cible ; c'est un régime d'interaction, pas une querelle de personnes, et quiconque y entre finit par y jouer les deux rôles.

Le cas donne à voir, daté et archivé, chaque phase de la boucle. Je le présente dans l'ordre des mécanismes, chaque pièce étant sourcée en note ; le lecteur est invité à vérifier — c'est tout l'intérêt d'un cas public.

Aucun des mécanismes décrits ici n'est établi par ce cas : tous ont été décrits ailleurs, sur d'autres terrains, des décennies avant lui. Ce que le cas apporte, c'est la boucle entière, datée, dans un seul projet — et c'est pour cela qu'il faut la documenter : la boucle qui s'est refermée ici se refermera de la même façon sur n'importe quel projet qui fonctionne selon les mêmes règles, et on ne peut la reconnaître à temps que si quelqu'un l'a montrée une fois, pièces en main. Et il faut le lire pour ce qu'il est : non le procès d'un homme par un groupe, ni d'un groupe par un homme, mais le fonctionnement d'un système — qui a fait taire bien d'autres voix que la mienne ; la mienne était seulement la plus sonore.

Phase un : les voix internes meurent. Le chapitre « Écouter sans obéir » a raconté le « tais-toi et répare », et donné son nom de littérature : le silence organisationnel. Il faut d'abord établir que ce silence n'a rien de personnel, car il a une histoire, antérieure à moi. Dès 2016 — six ans avant mon départ —, un utilisateur demandait sur le forum de la communauté qu'on stabilise l'existant avant d'empiler les fonctionnalités : on lui répondit que les utilisateurs qui ne contribuent pas sont « un fardeau, pas un atout », que ses arguments relevaient « de l'ignorance », et le fil fut fermé le lendemain — la question des besoins, elle, ne fut jamais traitée[28]. Le registre du forum tient l'inventaire de la suite : seize fermetures de fils pour « échauffement » de 2016 à 2026[29] — dont, en 2021, un utilisateur suspendu deux mois pour avoir demandé si les priorités de développement étaient les bonnes — et une formule maison, « personne ne vous force à l'utiliser », attestée dès 2018[30].

Puis vint l'affaire des filtres de collection, et elle vaut d'être racontée en détail, parce qu'on y voit toutes les voix mourir ensemble, en trois semaines, dates à l'appui. Avril 2022 : la refonte de l'outil central de tri des photographies — 7 600 lignes — est fusionnée au terme d'une discussion à 8 participants, et le module qu'elle remplace est supprimé du code le jour même ; la solution simple qu'un autre contributeur avait proposée en février 2022 — dix fois plus courte, et bien plus élégante à mon avis — avait été écrasée au passage, et la question du doublon, posée par écrit, ne reçut jamais aucune réponse[31]. En quelques semaines, une douzaine de tickets s'opposent à la refonte[32]. Le jour même de la découverte, sur le salon du projet, le mainteneur de la documentation écrivait déjà que la refonte ressemblait « à un développeur qui gratte sa propre démangeaison »[33] ; le ticket le plus fourni n'est pas de moi : c'est lui qui l'ouvre, et il réunit 14 participants et 89 messages — un développeur du cœur y juge le nouveau design ni lisible ni intuitif, et le responsable de la communauté y demande le retour à l'interface précédente — et il est toujours ouvert en 2026. Ma proposition de révocation, mise au vote, recueille 6 voix contre 4 : le mainteneur la refuse — « je suis tout à fait opposé à cette PR », puis, dans la discussion voisine, au contributeur qui reste insatisfait, « je ne peux pas vous forcer à être heureux :) », et enfin la doctrine : « je suis toujours pour aller de l'avant ». Les motifs opposés aux critiques valent d'être collectionnés, car chacun servit contre plusieurs voix à la fois : les mois de travail investis — le coût irrécupérable du chapitre « Décider, puis tenir », invoqué par écrit ; « les gens n'aiment pas le changement », haine de GNOME 3 à l'appui ; « les enfants crient jusqu'à obtenir ce qu'ils veulent, pas les adultes »

— la douleur rapportée traitée par la condescendance comme un enfantillage ; « restez constructif » — qu'y a-t-il de constructif dans le fait de silencier l'opinion majoritaire pour défendre un changement que personne n'avait demandé ? Une concession fut fusionnée en 77 minutes — la préférence qui restaure l'ancienne interface — quand la révocation majoritaire était morte en une semaine ; et les notes de version de Darktable 4.0 annoncèrent la refonte sans mentionner que la préférence de repli existait. Trois semaines plus tard, le signalement du mainteneur de la documentation fut fermé sur un correctif de façade — une infobulle — contre l'avis de son auteur, qui le rouvrira en décembre. En août, le modérateur du forum proposa d'instituer une revue de conception pour les grands changements, en citant nommément cette affaire ; 7 approbations, fermé en 5 jours : « très bien, nous faisons donc déjà de notre mieux »[34]. En décembre, un développeur du cœur dressa le bilan de l'intérieur — « la plus grande erreur a été de ne pas garder l'ancienne interface par défaut » —, et le principal défenseur de la refonte lui donna raison ; en 2026, sur le même fil toujours ouvert, un testeur de longue date résume : « puis l'auteur du fil et moi avons été réduits au silence :-) »[35].

Voilà pourquoi cette affaire vaut mieux qu'un grief. Sur un choix central, presque toutes les catégories de voix — utilisateurs, testeurs, contributeurs, un développeur du cœur, le mainteneur de la documentation, le modérateur du forum lui-même — ont demandé la même chose, au même moment, avec un vote majoritaire à l'appui ; et rien de tout cela n'a pesé face à un seul homme pressé d'« aller de l'avant ». J'étais l'une de ces voix — la

plus sonore. La parole qui meurt dans cette phase n'est pas la mienne : c'est celle de tout le monde. Mais je suis celui par qui le problème a été porté à la connaissance des utilisateurs.

Phase deux : l'exit. Quand la voix est morte, disait Hirschman, il reste la défection — et dans le logiciel libre, la défection a un nom technique : le fork[36]. J'ai forké. Et c'est ici que la chronologie, reconstituée date par date, est intéressante : l'hostilité publique n'a pas suivi mes critiques — elle a suivi mon départ. Jusqu'au début de 2022, le forum de la communauté me couvrait de remerciements ; les premières attaques publiques se concentrent dans les semaines mêmes de la bifurcation (mai 2022) et de mon départ des canaux du projet, fin juin 2022 ; et mon grand article critique est postérieur de 8 mois à ce retournement — il est arrivé dans un champ déjà hostile, il ne l'a pas créé. La semaine même de mon départ, le salon du projet m'approuvait d'ailleurs encore : un habitué me remerciait « pour le travail et pour le coup de gueule », un autre jugeait la nouvelle interface « préoccupante » ; sur le forum public, un troisième défendait mon droit de dire les choses crûment[37]. Ce que le cercle a puni en premier, ce ne sont pas des arguments : c'est la défection. La loyauté trahie, pas la thèse défendue. Les registres comptent les défections silencieuses qui ont suivi la mienne : le chapitre « Le travailleur invisible » a mesuré la fonte — commits, propositions, relectures — de ceux qui sont partis sans rien dire. Et la chronologie porte une seconde leçon, plus fine, car mes critiques n'avaient pas toutes le même registre. Ce qui a coïncidé avec le retournement, c'est une vidéo de colère, la *moins* technique de mes critiques : des régressions d'ergonomie, dites crûment — et c'est le langage qu'on a retenu,

au point d'en retirer le lien du forum. La version la plus rigoureuse est venue 8 mois plus tard : l'article-fleuve, sourcé, chiffré, exemples à l'appui — construit précisément pour être difficile à contrer. Il n'a rien changé, ni en bien ni en mal : le sentiment était déjà retourné, et le fond n'a jamais reçu de réponse de fond. Il faut peser ce que cette asymétrie établit : si le *contenu* de la critique avait été la variable, la critique la plus solide aurait produit la réaction la plus forte — elle n'en a produit aucune. La réaction suivait deux autres variables, la forme et la défection, et aucune des deux n'est un argument[38].

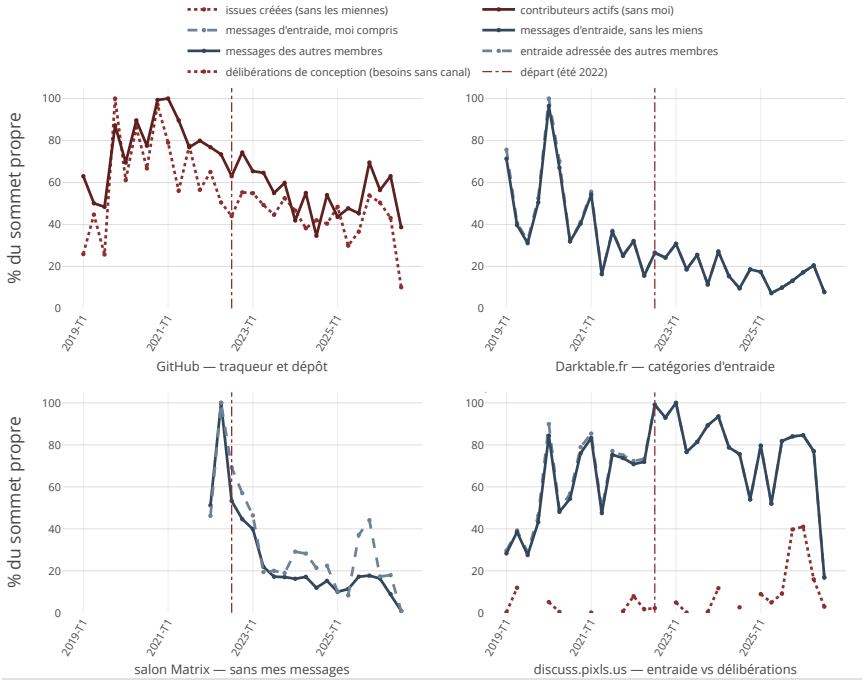
La chronologie dit quand le sentiment s'est retourné ; le journal du salon — le pont Matrix du canal IRC, dont j'ai depuis obtenu l'export — dit ce qu'il était avant de se retourner. On m'appréciait comme on apprécie un fournisseur : tant que je livrais des modules neufs et corrigeais des bugs, l'estime était sans réserve — les fils de remerciements suivent le calendrier de mes livraisons. La seule fois où j'ai dit la pénibilité de la charge dans le salon du projet — que je n'en pouvais plus de réparer ce que d'autres cassaient, souvent des choses que j'avais déjà réparées —, personne n'a proposé de me soulager : on a relevé le ton, et la réponse tient au chapitre « Écouter sans obéir » en 3 messages — trouve, répare, arrête de te plaindre[33]. L'affection était indexée sur le flux de livraison ; la personne sous le flux n'était le problème de personne. Et il faut nommer ce que cette boucle d'usure — réparer, voir casser, re-réparer — signale techniquement : un projet sans protocole de test ni de validation interne pour contenir les régressions. Non par manque de bras — le chapitre « Livrer, pas larguer » en a compté jusqu'à 130 par an

— mais par choix : les garde-fous, comme les critères d'acceptation du chapitre « Ce que l'entre-soi produit », auraient « nui au plaisir des développeurs ». Le confort du travailleur bénévole s'arrête là où commence celui des autres.

Phase 3 : la réécriture. Quatre ans plus tard, la même communauté raconte une autre histoire, et l'on peut documenter la dérive sur un même locuteur. Été 2022, le modérateur du forum : « personne n'a été rejeté ; il a demandé à partir et j'ai accédé à sa demande ». Été 2026, le même : « c'est notre faute — la mienne en particulier — d'avoir toléré un si mauvais comportement pendant si longtemps », tandis qu'autour de lui on ajoute que la communauté « prospère depuis qu'il est parti »[39]. « Il a choisi de partir » est devenu « nous l'avons enduré trop longtemps » : le récit a migré de mon libre choix vers leur patience héroïque, sans jamais passer par la case « pourquoi est-il parti ? ». La psychologie sociale n'attendait rien d'autre : la dissonance se réduit, et la mémoire collective se reconstruit au service du présent du groupe[40]. Deux détails donnent la mesure de la reconstruction : en 2022 même, un utilisateur s'étonnait publiquement d'entendre « bon débarras » autour du départ de celui qui avait « porté Darktable à un niveau entièrement nouveau »[41] — le climat que le récit officiel niait était donc audible en temps réel ; et en 2023, un tiers décrivait la discussion en cours comme ayant « toute l'apparence d'une attaque en meute »[42] — le terme n'est pas qu'imagé : la sociologie du travail nomme *mobbing* ce harcèlement de groupe qui converge sur une cible et finit par l'exclure, en 4 phases dont la dernière est

l'éviction[43]. Le cercle réécrit d'autant plus librement que les témoins contemporains ont, eux aussi, été archivés.

Or le récit du renouveau — la communauté « prospère depuis qu'il est parti » — affirme un fait, et un fait se teste. Aucun registre ne le confirme ; tous montrent l'inverse. Les contributeurs actifs du dépôt culminent en 2020-2021, mes années les plus actives, puis fondent — le chapitre « Livrer, pas larguer » en a donné les courbes. Le salon Matrix/IRC de discussion du projet, *mes messages exclus du compte*, perd la moitié de son activité dans les 6 mois qui suivent mon départ, et les 4 cinquièmes en 18 mois. L'entraide qui s'y pratiquait — les réponses adressées d'un membre à un autre — tombe de 4 900 messages l'année de ma présence à 1 600 l'année suivante, et n'est jamais remontée[44]. Corrélation n'est pas causalité, et je ne m'attribue pas l'activité qui avait lieu autour de moi pendant mes années actives. Mais « prospère » est une affirmation de fait, et aucune mesure ne la soutient : sur toutes les métriques, la maison est plus vide qu'avant, et l'assistance utilisateur est en net déclin partout, ou au mieux stagnante.



Le récit du renouveau, testé sur 4 lieux, 2019-2026. *Tout est compté en messages ou en événements datés à leur vrai trimestre ; chaque série est en pourcentage du sommet propre de sa série de référence ; le tireté marque le départ (été 2022) ; 2026-T3 est partiel et exclu du choix des sommets. Les registres :*

- ***GitHub*** *: issues créées et contributeurs actifs — mes signalements et mes commits sont exclus des comptes.*
- ***Darktable.fr*** *: messages des catégories d'entraide seulement (excluant les présentations obligatoires des nouveaux membres), avec puis sans mes messages.*
- ***Salon Matrix/IRC*** *(ouvert à partir de 2022) : total des messages et réduction aux messages d'entraide adressés à*

d'autres membres, sans les miens.

- ***discuss.pixls.us*** *: messages d'entraide, aux mêmes conventions, moins les annonces et préversions de modules et les délibérations de conception ; la ligne pointillée rouge isole ces délibérations — terminologie, refontes, fils du conflit — qui ne sont pas de l'entraide mais des besoins enregistrés sans canal.*

Deux lectures, développées dans le corps du texte :

- ***l'échelonnement des sommets*** *— le traqueur de bugs et l'entraide francophone culminent en 2019-2020, mes années les plus actives ; le forum francophone décroche dès 2021, l'année où ma participation s'y effondre ; les chats Matrix/IRC du projet culminent en 2022 puis tombent de manière spectaculaire sans rebond, en volume total comme en messages d'entraide.*
- ***le découplage*** *— l'entraide de pixls.us culmine en 2023 puis s'effrite : la communauté d'usagers survit, elle ne croît plus, pendant que les lieux du projet se vident ; les délibérations, quasi nulles avant 2025, explosent ensuite (ligne pointillée rouge), avec les annonces de modules, retranchées du compte et décomposées ci-dessous.*

Limites assumées : le volume de messages ne mesure pas leur qualité — je ne connais pas de métrique défendable pour l'en séparer — et le test ne porte pas sur la cause du déclin, mais sur l'affirmation d'un mieux, qu'aucun registre du projet ne mesure.

Il faut prévenir ici une mauvaise lecture, car les volumes bruts du forum racontent le contraire à qui ne les décompose pas : en apparence, la catégorie Darktable de pixls n'a jamais autant écrit

qu'en 2025-2026. Décomposée, la prospérité change de nature. L'entraide proprement dite — répondre au problème de quelqu'un — a culminé en 2023, portée par la vague des transfuges de Lightroom qui demandaient leur chemin, et s'effrite doucement depuis. Ce sommet est la dernière vague d'utilisateurs ordinaires — des photographes qui arrivaient et cherchaient leur chemin. Aucune vague n'a suivi. Le déclin de l'entraide n'est donc pas un effondrement du stock, c'est un tarissement du flux : les anciens restent et posent moins de questions, les nouveaux ne viennent plus — et tout ce qui croît à leur place est l'activité des ingénieurs-amateurs. Le mot juste est démographique : la base d'utilisateurs se *gentrifie* — mes propres enquêtes le mesuraient déjà, la part des programmeurs parmi les utilisateurs passant de 46 à 55 % entre 2020 et 2022 (chapitre « Ce que l'entre-soi produit »). Ce qui croît est double. Les annonces et préversions de modules, d'abord, passées de 3 à 6 % des messages avant 2025 à un tiers en 2026 : les utilisateurs-ingénieurs ont découvert l'intelligence artificielle et construisent leurs jouets — masques neuronaux, débruitage par réseau, modules de contraste en préversion — jusqu'à lasser le forum lui-même : en mars 2026, à un jour d'intervalle, deux fils s'y ouvrent — « Inquiets des modules écrits par LLM », 265 messages lancés par un contributeur du projet devant « deux propositions de modules en une semaine » générées par intelligence artificielle, puis « De la prolifération des modules »[45]. Les délibérations de conception, ensuite — la terminologie d'AgX (un nouveau module en construction), les refontes d'interface, « Darktable doit-il rester sans IA ? » — quasi inexistantes avant 2025, puis en explosion : près de 1 900 messages en 2025, un rythme de plus de 3 000 en

2026 — 10 fois le niveau de l'année du conflit. Le design par comité a quitté GitHub et finit en forum, comme les fils de 2022 l'annonçaient.

Ces fils de délibérations, il faut les lire deux fois. Comme méthode, ces délibérations sont le chapitre « Le comité n'inventera rien » en acte : des choix de conception soumis à la conversation, sans critères ni mandat. Mais comme matière, elles sont des besoins d'utilisateurs — enregistrés sans canal, faute du cahier des charges qui les recueillerait, exactement comme les *issues* de GitHub enregistraient des besoins avant de mourir sous le robot. Car l'entraide et le signalement mesurent la même chose — le soin porté aux problèmes d'autrui, problèmes de compréhension de l'outil, et problèmes de stabilité ou d'utilisabilité de l'application. Que les deux s'étiolent ensemble pendant que prospèrent les jouets et les débats sans mandat donne la trajectoire complète : le projet dérive en fablab d'utilisateurs-ingénieurs, toujours plus loin du logiciel « par des photographes, pour des photographes » qu'il prétend être — les besoins des photographes, eux, continuent de s'écrire ; simplement, plus personne n'est chargé de les lire[46].

La réécriture a une police, et elle s'exerce sur deux terrains. Le premier est le statut : parti, je n'avais plus le droit d'être discuté — « cela ne vaut pas la peine de discuter de gens qui ne sont pas présents ici », « si vous voulez parler avec son développeur, vous ne pouvez pas le faire ici » ; en 2025, un contributeur proposa d'appliquer « une version de la loi de Godwin » à quiconque rouvrirait le sujet[47]. Mes critiques publiées après le départ — la vidéo, puis l'article — furent ainsi disqualifiées deux fois : sur le

ton, et sur le statut, puisque je n'étais plus des leurs, donc plus fondé à parler — on m'opposa aussi que je ne savais pas « travailler en groupe » (critique apparue après mon départ, jamais formulée avant), et qu'il fallait « rester constructif » : l'esprit de corps invoqué comme un secret à garder, dans un projet où tout était déjà public. Le second terrain est le contenu, et le refus y fut érigé en politique, par écrit : un modérateur du forum communautaire déclara que « les mérites techniques de son code, de ses modules, de sa vision ou de quoi que ce soit d'autre pâlissent en comparaison de son comportement toxique », et le mainteneur principal du projet formula la règle : tant que l'intéressé n'apprendrait pas à parler aux autres, il ne serait « plus accordé une seule pensée à ce qu'il a à dire, point final »[48]. C'est la violence épistémique décrite plus haut, portée à son terme et couchée par écrit : assigner au locuteur un déficit de crédibilité tel que le contenu n'a plus jamais à être évalué. Notez ce que la politique déclare exactement : non pas « il a tort », mais « on ne lira pas ». Le cercle ne réfute pas ; il révoque le droit d'être lu — la police du ton décrite plus haut, cette fois couchée par écrit et érigée en politique officielle. Et quand j'ai publié, pièces à l'appui, le diagnostic technique de la dégradation — volumes de code, scores de complexité, extraits —, la réponse mesurable a eu une structure remarquable : les faits d'implémentation furent parfois discutés ; les questions de conception furent réglées par l'autorité ; et les mesures de maintenabilité ne reçurent aucune réponse, jamais — sur tout le registre du forum, les mots « cyclomatique », « complexité cognitive » et « SonarCloud » n'apparaissent pas une seule fois[49]. Personne, en quatre ans, n'a répondu à un chiffre de complexité par un chiffre de complexité, à

une inquiétude sur la maintenabilité du projet à moyen/long terme par une action tangible censée la protéger. La culture qui se dit directe aura donc évité, des années durant, les questions les plus directes qu'on lui ait posées — tout en restant parfaitement directe sur les personnes. Direct sur l'homme, évasif sur le code : cela ne s'appelle pas de la franchise, cela s'appelle du ciblage.

Reste ma part — et ce n'est pas une quatrième phase : c'est une boucle qui courait sous les 3 autres, du premier signalement à la rupture. Mon registre s'est durci, et le point de rupture est daté dans les archives IRC/Matrix : le 7 mai 2022. Après des années d'un registre parfaitement civil sur le dépôt du projet, tout bascula le jour même où, en rebasant mes branches, je découvris la refonte des filtres de collection que je signalai alors dans un rapport de bug (tant le design était improbable et l'idée que ce fût volontaire était incongrue). Le journal du salon Matrix du projet documente ce jour-là heure par heure — la découverte à 12 h 57, le rapport de bug à 13 h 26 — et il documente tout autant la suite : la perte d'une patience qui avait déjà été mise à mal depuis fin 2021[33]. Je ne suis pas de ceux qui subissent en silence. Mais il faut nommer correctement ce qui s'est durci. La colère est venue contre des paralogismes assumés — faux, expéditifs, prononcés depuis des positions de pouvoir, inattaquables et néanmoins irresponsables — que j'ai refusé de tolérer davantage.

Les historiques de mes communications sur Github sont publics et ne montrent que des échanges courtois avec mes collaborateurs avant 2022. Les journaux IRC/Matrix sont plus épicés, mais c'est le style du lieu, et même là : aucune attaque contre quelqu'un avant fin 2021. Pire, entre 2019 et 2021,

j'organisais une levée de fonds dans la communauté, à l'approche Noël, pour expédier au mainteneur — qui refusait les dons en monnaie — un panier de victuailles artisanales, pour lui et sa famille pendant les fêtes. C'est sur le forum pixls.us qu'on peut trouver quelques emportements, en face d'utilisateurs qui critiquaient mon travail sans chercher à comprendre les problèmes qu'il résolvait, ou s'obstinaient à ignorer mes conseils tout en se plaignant du résultat contraire à leurs attentes. L'exposition permanente, dans une transaction à un contre plusieurs, où chacun peut se permettre une opinion sans avoir étudié le dossier, impose de tenir son pré carré avec fermeté pour dissuader les suivants de s'y risquer. La tranquilité est à ce prix.

Cependant, quand des années de politesse n'ont rien protégé, rien amélioré, mais n'ont au contraire été qu'une complicité muette avec le saccage de l'outil pour le plaisir des développeurs, cesser de feutrer cette colère est un refus : un acte de résistance au sens posé plus haut — la violence visible de celui à qui il ne reste que les moyens visibles, parce que la situation est inacceptable selon toutes les métriques applicables. La certitude que la situation était définitivement bloquée, hors de toute réparation possible fut acquise pièce par pièce au cours du temps. Entre 2020 et 2022, j'ai vu une nouvelle habitude émerger : celle de voir des fonctionnalités qui marchaient raisonnablement être remplacées, une à une, par des horreurs de sur-ingénierie sans qu'aucun problème n'ait été posé — le crime du chapitre « Poser le problème », en série. Un vétéran de 10 ans l'a dit publiquement dans les mêmes termes que moi : le meilleur Darktable qu'il ait utilisé était le 1.4, et chaque version depuis n'a fait que se

dégrader[50] — un déclin de réactivité dont je trouverai plus tard la cause, en nettoyant le code dans Ansel : l'accumulation d'erreurs de programmation ordinaires dans le code d'interface, directement imputable à une absence d'hygiène de travail. Et quand je signalais ces régressions, on me répondait par des paralogismes documentés comme tels : pour défendre les filtres de collection, on m'opposa des mois de travail investi — la fallacie du coût irrécupérable du chapitre « Décider, puis tenir », invoqué comme argument, par écrit, dans le traqueur du projet[51]. Un paralogisme courtois, prononcé depuis une position de veto, reste un paralogisme armé — la civilité du ton n'en dissimule que mieux la dangerosité. Quand on a acquis, pièces en main, la certitude que la politesse ne protège plus le projet, on change d'objectif : je n'essayais plus de convaincre, j'essayais de *dissuader* — de rendre coûteux le fait de dégrader encore. Dans le far-west décrit plus haut, c'était l'arme qui me restait. Car il faut mesurer l'état du système : le nombre, je l'avais — le vote majoritaire, un développeur du cœur, le mainteneur de la documentation, le modérateur du forum lui-même : la première phase a compté les voix. Et c'est pire ainsi : dans un far-west, le nombre ne tranche rien — tranchent les droits d'accès et l'usure. Leur temps libre, je ne l'avais pas ; et la rationalité avait été perdue.

Il faut dire aussi ce que mes critiques de l'époque n'étaient pas, car c'est une pièce du dossier : épuisé, en surchauffe — le chapitre « Le travailleur invisible » a décrit ce régime —, je m'en suis tenu, dans ce que je publiais, aux faits techniques : volumes de code, scores de complexité cyclomatique et cognitive, commandes

reproductibles[49]. Une partie de mon diagnostic restait de l'intuition, et je le savais ; l'analyse psychologique et sociale, celle de ce livre, a attendu la distance — et un endroit sûr pour la mener. Le fond n'a pas changé ; il est simplement, désormais, instrumenté.

Ce durcissement fut un acte de résistance, et aussi une stratégie de dissuasion ; jugé comme stratégie, son bilan, établi avec les mêmes méthodes que le reste, tient en deux temps. Le premier est sans appel : la dissuasion n'a rien dissuadé — les horreurs ont été fusionnées quand même, car dans un far-west ce sont les droits d'accès qui tranchent, pas les coups de gueule. J'avais moi aussi les droits de commit sur le dépôt, et je les ai conservés jusqu'en 2026 : la capacité matérielle de tout annuler, refonte par refonte, jusqu'à l'état antérieur, est restée entre mes mains pendant 4 ans. Je ne m'en suis jamais servi. Annuler ces refontes aurait imposé ma décision par le même far-west que je dénonce — les droits d'accès à la place des raisons — et l'on ne sauve pas de lui-même un projet dont c'est le mode de gouvernement. Aucune issue productive, constructive, n'existait dans ce système. La seule mesure à la hauteur du problème eût été de démettre l'équipe — dans une structure où les utilisateurs ont un pouvoir de vote, c'est une option ; ici, personne ne détient ce pouvoir, le chapitre « Le comité n'inventera rien » a dit pourquoi. Et partir plus tôt n'était pas davantage une issue : mon travail était enchevêtré dans le leur, sans moyen simple de l'en extraire — la défection coûte d'autant plus cher qu'on a plus donné, et c'est pourquoi elle vient si tard.

Le second temps du bilan est venu plus tard, et il change la nature du premier. Appeler ma stratégie un « échec » supposerait qu'une autre pouvait réussir. Or rien de ce que j'ai signalé n'a été corrigé depuis — la fin de ce chapitre en fera le compte — et aucune stratégie n'a gagné : la politesse a perdu, le vote majoritaire a perdu, la procédure a perdu — la première phase les a comptés. La partie était verrouillée. Quand toutes les stratégies échouent, le problème n'est pas la stratégie.

La preuve la plus nette est venue en 2026, et elle ne m'appartient pas. Lors de la controverse sur l'entrée de l'intelligence artificielle dans Darktable, les deux hommes qui avaient tenu la ligne contre moi — le modérateur du forum, le mainteneur de la documentation — ont quitté le projet à leur tour. Le premier l'a écrit noir sur blanc, au mainteneur : « j'ai essayé de faire entendre mon avis, et le mainteneur m'a répondu que je menaçais le projet… je suis parti à cause de votre traitement et de vos actes, directement » ; il détaille ses griefs — un vote des « développeurs du cœur » convoqué en excluant les autres mainteneurs, ses contributions traitées comme sans valeur, une accusation jamais suivie d'excuses, puis le silence[52]. Le second s'est retiré sans accuser personne — « il valait mieux pour le projet que je m'écarte » — mais les registres datent son arrêt : ses fusions de documentation s'interrompent la semaine même où la décision passa outre l'avis des mainteneurs de la documentation, exclusion que le mainteneur du projet décrivit lui-même : « les mainteneurs de la doc étaient contre — bien ; maintenant je voulais savoir ce qu'en pensaient les développeurs »[52]. Et le jour de leur départ, le récit officiel fut le même que pour le mien :

« Mica et Elstoc ne sont plus intéressés par la maintenance de la documentation » — *plus intéressés* ; le premier répondit que c'était « ajouter l'insulte à la blessure », la réécriture n'avait même pas attendu[52]. Pas un mot du deuxième.

L'ironie n'a pas besoin d'être savourée, car elle a valeur de preuve. Ces deux hommes m'ont reproché ce que je suis, ce que je dis et comment je le dis ; ils ont fermé mes fils, retiré mes liens, fait de ma « toxicité » la politique officielle. Puis ils ont refait, des années plus tard et par leur propre chemin, exactement mon diagnostic — un pouvoir sans partage, des décisions sans consultation, des contributions sans valeur aux yeux d'un seul — et ils en ont tiré la même conséquence que moi : le départ. Le fond est donc désormais commun ; seul le reproche de forme est resté. C'est la démonstration finale de la police du ton : quand on partage le verdict d'un homme et qu'on maintient sa condamnation, c'est qu'on ne l'a jamais condamné que pour sa manière de le dire. On les attendrait dans la résistance à laquelle ils donnent désormais raison ; ils gardent la porte à la place — l'un modère toujours le forum, où il a lui-même fermé les fils de la controverse ; l'autre s'est tu « pour le bien du projet ». Peut-être est-ce la loyauté de Hirschman, dernier mécanisme du cercle ; peut-être ce que coûterait l'aveu — admettre que le banni avait raison obligerait à rejuger le bannissement. Je n'arbitre pas entre les deux : les pièces suffisent.

Quant à la voix durcie, elle a fait une chose : dire tout haut des inquiétudes que je n'étais pas seul à porter — la première phase les a comptées, et le journal du salon les date. Mais le même journal mesure pourquoi tout cet accord n'a rien produit : celui

qui décidait n'était pas dans la pièce. Le fusionneur qui signe 98 % des fusions y a écrit 36 messages en 4 ans et demi — l'approbation se dépensait, littéralement, hors de portée de la seule personne qu'il aurait fallu convaincre, et ceux qui approuvaient là n'allaient pas le redire sur le dépôt, où cela se décidait. L'accord d'arrière-salle n'était pas un défouloir par accident : c'était le seul registre où il ne coûtait rien, précisément parce qu'il n'y produisait rien. Ce phénomène a ses noms : la *spirale du silence* — qui se croit minoritaire se tait, et son silence grossit la majorité apparente qui fera taire le suivant — et la *falsification des préférences*, l'écart entretenu entre ce qu'on pense en privé et ce qu'on soutient en public[53].

La conséquence est cruelle pour le dissident public : l'unanimité qu'on lui oppose est en partie fabriquée par la peur, mais elle a *l'air* réelle, et chaque contestataire paraît donc isolé — donc excentrique, donc « toxique » — alors qu'il est parfois le porte-parole involontaire d'une opinion qui se cache. Un modérateur du forum affirmait que « les gens qui aiment [l'interface] ne le disent pas si fort » ; le registre dit mieux que l'inverse : ceux qui ne l'aimaient pas le disaient — la première phase les a comptés — mais un critique qu'on peut traiter un par un se congédie plus facilement qu'une majorité ne se laisse compter, et chaque congédiement rendait le suivant plus silencieux. Et la certitude ? Elle était exacte — à condition de définir le mot. Ce que j'appelais idiotie n'a jamais été l'incompétence de bas niveau : c'était l'incapacité à choisir les bonnes priorités — garder l'ergonomie alignée sur les conventions du bureau, simplifier le code au lieu de remplacer ce qui marche par de la sur-ingénierie — et

l'absence de compréhension structurelle de leur propre application : une résolution de problèmes en permanence sous le lampadaire du chapitre « La racine et le symptôme », là où c'est éclairé, jamais là où c'est cassé. Les contre-exemples du dossier ne réfutent pas ce diagnostic : ils le précisent — les engagements honnêtes qui s'y trouvent portent tous sur des faits d'implémentation, le seul niveau où le milieu s'engage ; les mesures de maintenabilité et les questions de conception, on l'a vu, n'ont jamais obtenu de réponse. Une seule fois dans tout le dossier, un argument d'architecture a été avancé pour justifier un choix d'interface — et il ne m'était pas même adressé : 3 ans après mon départ, dans une énième discussion d'interface sur le forum, le fusionneur principal expliquait à d'autres utilisateurs que le logiciel étant « à base de greffons », il était difficile d'en contrôler l'ordre à l'écran. L'argument a le mérite d'être vérifiable ; mesuré sur le code, il est faux — l'ordre des modules y est précisément la chose la plus centralement codée en dur de tout l'édifice, et mon propre fork, mesuré aux mêmes critères, reste le même monolithe, à peine atténué[54]. Une intelligence strictement locale : le défaut n'est pas neuronal, il est disciplinaire — savoir coder n'est pas savoir concevoir, et ce livre entier tient dans cet écart. Mon signalement était fondé ; resté des années sans réponse, il s'est durci en colère. La colère a fourni le prétexte pour ne plus me lire du tout. Et ne plus être lu ne me laissait que la colère. Chaque tour a resserré le suivant, jusqu'à la rupture. Un système qui ne laisse, contre sa propre dégradation, d'autre levier que l'invective — puis se sert de l'invective pour ne pas entendre parler de la dégradation — est un système sans mécanisme d'apprentissage. Et c'est cela, le défaut de conception.

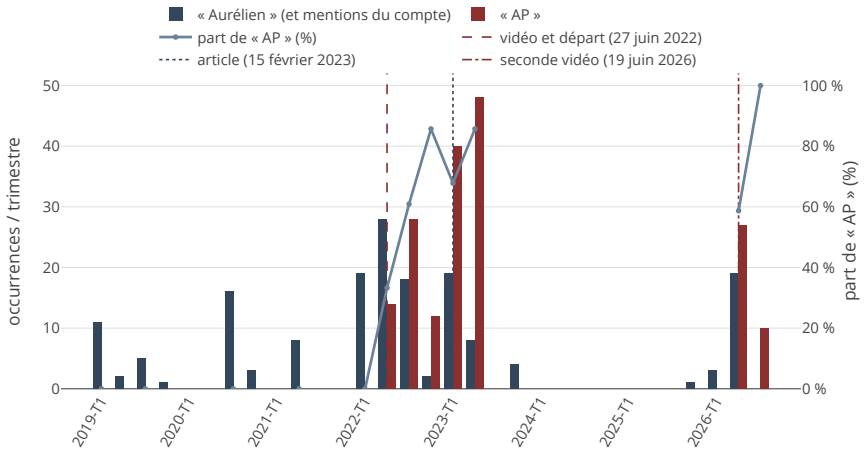
Quatre ans plus tard, le dossier est clos sur un point : le diagnostic était juste. Non parce que je l'affirme — parce que les registres le mesurent, et ce livre les a produits chapitre après chapitre. Les corrections déclinent et l'encours d'*issues* regonfle (« Livrer, pas larguer »); l'engagement fond des deux côtés, développeurs comme utilisateurs (« Le travailleur invisible »); le taux de refus des propositions est au plus bas de l'histoire du projet, pendant que le droit de fusionner s'est concentré en un seul homme — 98 % des fusions depuis 2019 — qui ne peut matériellement pas relire ce qu'il fusionne (« Le culte du cargo », « L'ambition non héroïque »); et les lieux du projet se vident pendant que le forum dérive en fablab — la maison vide, mesurée plus haut. Même la promesse constitutive est démentie par ses propres chiffres : le plaisir de programmer devait garantir les bras, et le chapitre « Travailler pour le travail » a montré le recrutement tari. Le projet n'est pas en bonne santé, à l'aune même de ses propres critères.

Il faut enfin dire qui est « ils », car ce mot a servi tout ce chapitre et il serait malhonnête de le laisser flotter. « Ils » n'est pas une foule : c'est un rôle, tenu par peu de monde — le fusionneur unique, qui décide et — la mesure en a été donnée plus haut — n'est pas dans la pièce où l'on parle; le noyau de modération du forum, qui ferme et étiquette; et une poignée d'habitués. Autour d'eux, la majorité assiste et les coulisses montrent qu'elle n'approuve pas toujours ce qu'elle laisse faire : pendant que le mainteneur défendait publiquement la maison, le mainteneur de la documentation écrivait en privé que l'interface actuelle est « simplement mauvaise » et qu'il ne supportait plus les disputes

sans fin avec lui, qui échouaient à prévenir la dégradation[55]. Un passant du forum a dit la structure mieux que la sociologie : « une seule personne ne peut pas créer une “atmosphère toxique” ; il y faut la participation et la complicité de plusieurs »[56]. Et le système immunitaire tourne encore, contre des données désormais établies : 33 fils ont instruit mon cas en mon absence ; on propose la loi de Godwin contre quiconque rouvre le dossier ; et en 2026 encore, la réponse officielle aux critiques est le souci de l’image — on pourrait « croire que nous sommes une bande d’idiots qui font un mauvais logiciel »[57]. Le diagnostic est confirmé ; il n’est toujours pas entendu. Un cercle fermé ne s’ouvre pas quand il a tort : c’est même à cela qu’on le reconnaît.

La signature finale du cercle fermé n’est pourtant ni l’expulsion ni la réécriture : c’est ce qu’il fait de la contribution de l’expulsé. Le dossier documente l’absorption sans le nom : mes correctifs repris dans le projet d’origine, au commit près ; mon estimation du chantier de fondations, refaite indépendamment 4 ans plus tard par un habitué, aboutissant aux mêmes ordres de grandeur, sans référence ; mon diagnostic global, approuvé aux deux tiers par un vétéran du projet[58]. Le cercle peut utiliser le travail de celui qu’il a banni ; il ne peut pas prononcer son nom. Ce n’est pas une image, c’est une mesure. J’ai compté les formes par lesquelles le forum me désigne, sur 1 931 messages couvrant 14 fils de 2019 à 2026 — ceux qui portent sur mon travail ou sur mon départ[59]. Trois ans durant, avant la rupture, on m’appelle par mon prénom : 65 occurrences, zéro initiale. La première occurrence de « AP » date du 28 juin 2022 — le lendemain de la vidéo — et dans les 48 heures, 7 intervenants distincts l’adoptent ;

sur l'année qui suit le départ, les initiales font près des deux tiers des désignations, et les titres mêmes des fils suivent : « Interview with AP » (été 2022), « What is it about Ansel and AP ? » (2023), « AP's new video on Ansel » (2026). Le mécanisme est daté au jour près, et il dit quelque chose de plus que la distance : tant que j'étais là, on me parlait ; à partir du départ, on ne pouvait plus que parler *de* moi — et pour parler de l'absent, le groupe n'a pas gardé le prénom, il a choisi le sigle. Au bout de la pente vient la troisième forme : « a previous developer », « that person is no longer here », « the person who designed » — la référence sans personne. Prénom, initiales, personne : la langue du groupe a suivi exactement le trajet de la réécriture. Ce n'est pas de l'ingratitude, c'est de la cohérence : créditer l'expulsé rouvrirait la question que la réécriture a close. Un cercle fermé se reconnaît à ceci : on peut y être d'accord avec le banni, à condition de ne pas donner la source.



Comment le forum me nomme, 2019-2026 : occurrences par trimestre du prénom (« Aurélien », y compris les mentions du compte — réécrites a posteriori en @anon41087856 lors de la suppression de mon compte, à ma demande) contre les initiales (« AP »), sur 1 829 messages de 14 fils consacrés à mon travail ou à mon départ — les miens exclus ; en trait continu, la part du sigle dans l'ensemble des désignations (axe de droite, tracée à partir de 5 mentions par trimestre). 3 ans à 0 % ; la première initiale apparaît le lendemain de la première vidéo (tirets rouges), la part passe à 61 % le trimestre suivant et ne redescend plus durablement — l'article de février 2023 (pointillé bleu) relance le volume, pas la forme, et la seconde vidéo (tirets-points rouges, 19 juin 2026) retrouve le même régime, jusqu'à 100 % sur le trimestre en cours. Les trimestres vides sont ceux où aucun fil du corpus n'est actif ; 2026-T3 est partiel (juillet).

La sociologie des sciences a un nom pour ce régime, et il vaut d'être invoqué exactement. Merton appelait *oblitération par incorporation* le destin normal des idées qui réussissent : leur source s'efface à mesure que leur contenu s'incorpore au savoir commun — on cesse de citer celui dont tout le monde répète les résultats[60]. Mais Merton décrivait un processus séculaire, impersonnel et sans hostilité. Le registre du projet en donne une version accélérée et datée, où l'effacement du nom et la réutilisation du travail avancent du même pas :

Année	Part du sigle dans les désignations (forum)	Réutilisations documentées, la même année
2022	45 % (zéro les 3 années précédentes)	mes vidéos recommandées en réponse aux questions de couleur (salon, 10 occurrences)
2023	74 %	« nous devrons peut-être les porter depuis Ansel » (janvier) ; vidéos toujours recommandées
2024	corpus presque muet	« envoyez-lui donc cette vidéo d'AP » (janvier)
2026	63 %, puis la totalité des dernières mentions	rétroportages de correctifs au commit près (avril) ; estimation du chantier GTK 4 reproduite sans référence ; vidéos encore citées

L'oblitération mertonienne prend des générations ; celle-ci a pris 4 ans, et elle a commencé le lendemain d'une vidéo. Ce n'est pas le cours naturel du savoir : c'est le cours naturel du savoir, sous pression.

Un cercle fermé ne se rouvre pas de l'intérieur par la seule bonne volonté : chaque mécanisme de ce chapitre — le tri par l'agressivité, la révocation du droit d'être lu, la réécriture, l'étouffement de l'accord — est précisément fait pour neutraliser la bonne volonté individuelle. Il faut de la structure — mais avant de bâtir la structure, il faut réapprendre ce qu'elle devra abriter : le conflit lui-même, qui n'est pas la maladie de ce chapitre mais sa matière première mal traitée. Se disputer n'est pas le problème ; se disputer sans règles l'est. C'est l'objet du chapitre suivant.

---

1. Ferreira, I. *et al.* (2021). « The "Shut the f**k up" Phenomenon : Characterizing Incivility in Open Source Code Review Discussions ». *Proceedings of the ACM on Human-Computer Interaction*, 5(CSCW2), art. 353 https://doi.org/10.1145/3479497 — 1 545 courriels de la Linux Kernel Mailing List. ↩

2. Tannen, D. (1998). *The Argument Culture*. New York : Random House https://openlibrary.org/works/OL15841541W — cadre culturel général (le discours comme combat rituel), appliqué ici au logiciel libre par analogie, non comme résultat d'étude sur lui. ↩

3. Letourneur, D. (2022). *On ne naît pas mec. Petit traité féministe sur les masculinités*. Paris : La Découverte, coll. « Zones » https://www.editionsladecouverte.fr/on_ne_nait_pas_mec-9782355221521 . La définition est reprise ici pour son

rendement, indépendamment du terrain sur lequel l'autrice la déploie : la violence comme retrait d'options. La restriction que j'y ajoute — le retrait doit être arbitraire, c'est-à-dire sans nécessité démontrée — est de moi, et elle est nécessaire : sans elle, toute règle deviendrait une violence. ↩

4. Andersson, L. M. et Pearson, C. M. (1999). « Tit for Tat ? The Spiraling Effect of Incivility in the Workplace ». *Academy of Management Review*, 24(3), 452-471 https://doi.org/10.2307/259136 , définition p. 457 : « low-intensity deviant behavior with ambiguous intent to harm the target, in violation of workplace norms for mutual respect » (notre traduction dans le corps du texte) — l'ambiguïté de l'intention est le trait définitoire, et ce qui rend la chose si difficile à sanctionner. ↩

5. Williams, K. D. (2007). « Ostracism ». *Annual Review of Psychology*, 58, 425-452 https://doi.org/10.1146/annurev.psych.58.110405.085641 — les 4 besoins fondamentaux menacés ; Eisenberger, N. I., Lieberman, M. D. et Williams, K. D. (2003). « Does Rejection Hurt ? An fMRI Study of Social Exclusion ». *Science*, 302(5643), 290-292 https://doi.org/10.1126/science.1089134 — l'exclusion sociale simulée (jeu de balle virtuel) active le cortex cingulaire antérieur, impliqué dans la douleur physique, proportionnellement à la détresse rapportée. ↩

6. Rochefort, C. (1971). « Définition de l'opprimé », texte de présentation de l'édition française de Valerie Solanas, *SCUM Manifesto* ; réédité en volume, Paris : éditions les vilains, 2021,

https://www.editionslesvilains.fr/livres/christianerochefort-definitiondelopprimee-texteintegral. « L'oppresseur n'entend pas ce que dit son opprimé comme un langage mais comme un bruit » ; et : « Pour l'oppresseur il n'y a pas d'oppression, forcément, mais un fait de nature. ». ↩ ↩

7. Fricker, M. (2007). *Epistemic Injustice : Power and the Ethics of Knowing*. Oxford : Oxford University Press https://doi.org/10.1093/acprof:oso/9780198237907.001.0001 (trad. fr. *Injustice épistémique*, Eliott, 2026). Fricker réserve le concept aux préjugés visant des groupes sociaux ; son application à un déficit de crédibilité individuellement construit est un emprunt de structure, signalé comme tel. ↩

8. Imran, M. M. & Sarker, J. (2025). « "Silent Is Not Actually Silent" : An Investigation of Toxicity on Bug Report Discussion ». *FSE 2025*, piste *Ideas, Visions and Reflections* https://arxiv.org/abs/2503.10072 — analyse qualitative de 203 fils de signalement GitHub, dont 81 toxiques. La toxicité y naît principalement de perceptions divergentes de la gravité et de la priorité du défaut, de frustrations non résolues envers l'outil, et de manquements à la communication professionnelle ; ces échanges « réduisent la probabilité d'issues actionnables, telles que le rattachement du signalement à une proposition de code ». Étude préliminaire, sur un échantillon volontairement enrichi en fils toxiques. ↩

9. Sarker, J., Turzo, A. K. & Bosu, A. (2025). « The Landscape of Toxicity : An Empirical Investigation of Toxicity on GitHub ». *Proceedings of the ACM on Software Engineering* (FSE 2025) https://doi.org/10.1145/3715744 — 2 828 projets

échantillonnés, classification automatique par un détecteur de toxicité spécialisé au génie logiciel. La popularité d'un projet est associée positivement à la prévalence de la toxicité, son taux de résolution des signalements négativement ; les auteurs de messages toxiques récidivent significativement plus, et deviennent eux-mêmes plus souvent cibles — ce qui rejoint le régime décrit ici, où chacun est tour à tour tireur et cible. Associations mesurées, non causalité établie. ↩

10. Johnson, A. (2017). « The Multiple Harms of Sea Lions », dans *Perspectives on Harmful Speech Online*. Cambridge, MA : Berkman Klein Center, Harvard University https://cyber.harvard.edu/publications/2017/08/harmfulspeech — le procédé « allie un questionnement persistant — souvent sur des points élémentaires, aisément vérifiables ailleurs, ou tangents — à une revendication bruyante de débat raisonnable » et « travaille à la fois à épuiser la patience, l'attention et l'effort de communication de la cible, et à la faire passer pour déraisonnable » (notre traduction). Le terme vient d'une bande dessinée : Malki, D. (2014). *Wondermark*, n° 1062, « The Terrible Sea Lion » https://wondermark.com/1k62/ . ↩

11. Deneault, A. (2015). *La Médiocratie*. Montréal : Lux https://openlibrary.org/works/OL27260738W — en français dans le texte : « Il n'y a eu aucune prise de la Bastille, rien de comparable à l'incendie du Reichstag, et l'Aurore n'a encore tiré aucun coup de feu. Pourtant, l'assaut a bel et bien été lancé et couronné de succès : les médiocres ont pris le

pouvoir » ; « Rangez ces ouvrages compliqués, les livres comptables feront l'affaire » (formulations vérifiées sur l'édition de l'éditeur). ↩

12. Bailey, A. (2018). « On Anger, Silence, and Epistemic Injustice ». *Royal Institute of Philosophy Supplement*, 84, 93-115 https://doi.org/10.1017/s1358246118000565 ; Cloud, D. L. (2021). « The Rhetoric of Civility as Soft Repression », in *Civility, Free Speech, and Academic Freedom in Higher Education*. Londres : Routledge. https://doi.org/10.4324/9780429282041-9 ↩

13. Sur la construction de la folie comme « maladie féminine » et la pathologisation psychiatrique de la déviance des femmes : Showalter, E. (1985). *The Female Malady : Women, Madness, and English Culture, 1830-1980*. New York : Pantheon. https://openlibrary.org/books/OL3023442M/The_female_malady Sur le congédiement de la colère des femmes par le diagnostic — écarter le contenu en qualifiant la forme : Chemaly, S. (2018). *Rage Becomes Her : The Power of Women's Anger*. New York : Atria. https://www.simonandschuster.com/books/Rage-Becomes-Her/Soraya-Chemaly/9781501189562 ↩

14. Lorde, A. (1981). « The Uses of Anger : Women Responding to Racism », discours à la National Women's Studies Association, juin 1981 ; repris dans *Sister Outsider*. Trumansburg, NY : The Crossing Press, 1984, p. 124-133 https://archive.org/details/sisteroutsideresoolord : « Tell me how you feel but don't say it too harshly or I cannot hear you. But is it my manner that keeps her from hearing, or the

threat of a message that her life may change ? » ; « Anger is loaded with information and energy » (notre traduction dans le corps du texte ; trad. fr. publiée : *Sister Outsider*, Genève : Mamamélis, 2003, essai « De l'usage de la colère » — à collationner avant impression). Sur la police du ton comme silenciation : Bailey, A. (2018). « On Anger, Silence, and Epistemic Injustice ». *Royal Institute of Philosophy Supplement*, 84, 93-115 https://doi.org/10.1017/s1358246118000565 . ↩

15. L'erreur fondamentale d'attribution : Ross, L. (1977). « The Intuitive Psychologist and His Shortcomings ». *Advances in Experimental Social Psychology*, 10, 173-220 DOI ; l'asymétrie acteur-observateur : Jones, E. E. et Nisbett, R. E. (1971). *The Actor and the Observer*. Morristown, NJ : General Learning Press https://openlibrary.org/works/OL10486732W . Malle, B. F. (2006). « The Actor-Observer Asymmetry in Attribution : A (Surprising) Meta-Analysis ». *Psychological Bulletin*, 132(6), 895-919 https://doi.org/10.1037/0033-2909.132.6.895 — sur 173 études, l'asymétrie générale est proche de zéro, mais elle subsiste pour les événements négatifs (d ≈ 0,24, ramené vers 0,11 après correction du biais de publication) et s'inverse pour les positifs : un motif d'auto-complaisance — qui est précisément le régime des distributions de blâme discutées ici. ↩

16. Becker, H. S. (1963). *Outsiders : Studies in the Sociology of Deviance*. New York : Free Press of Glencoe, p. 9 https://openlibrary.org/works/OL44024314W ; trad. fr. de J.-P. Briand et J.-M. Chapoulie, *Outsiders. Études de sociologie*

de la déviance. Paris : Métailié, 1985, p. 32-33 — les guillemets suivent la traduction publiée (à collationner sur l'édition papier avant impression). ↩

17. Baron, R. A. (1988). « Negative Effects of Destructive Criticism : Impact on Conflict, Self-Efficacy, and Task Performance ». *Journal of Applied Psychology*, 73(2), p. 199-207 https://doi.org/10.1037/0021-9010.73.2.199 — la critique destructrice y est définie par 3 traits : non spécifique, inconsidérée dans le ton, et imputant la mauvaise performance à des causes *internes* plutôt que situationnelles. Les destinataires rapportent davantage de colère et de tension, et déclarent qu'ils traiteront les désaccords ultérieurs par la résistance ou l'évitement plutôt que par la collaboration ou le compromis. C'est le construit que la recherche en génie logiciel a repris pour mesurer les revues de code destructrices, citées plus loin. ↩ ↩

18. Torvalds, L. (16 septembre 2018). « Linux 4.19-rc4 released, an apology, and a maintainership note », liste de diffusion du noyau Linux https://lkml.org/lkml/2018/9/16/167 : « I need to change some of my behavior, and I want to apologize to the people that my personal behavior hurt and possibly drove away from kernel development entirely. » Le projet remplace dans la foulée son *Code of Conflict* par un code de conduite dérivé du *Contributor Covenant* — « Code, conflict, and conduct », *LWN.net*, septembre 2018 https://lwn.net/Articles/765108/ ; texte en vigueur : Contributor Covenant Code of Conduct https://docs.kernel.org/process/code-of-conduct.html. ↩

19. Tourani, P., Adams, B. & Serebrenik, A. (2017). « Code of Conduct in Open Source Projects ». *IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)* https://doi.org/10.1109/SANER.2017.7884606 — première étude empirique des codes de conduite du logiciel libre : ils servent autant à prévenir qu'à sanctionner, mais leur application est inégale, et rien n'établit qu'à eux seuls ils facilitent l'arrivée de nouveaux venus. ↩

20. Hanika, J. & Ellinghaus, T. (2 juin 2019). « 10 years of Darktable », *Libre Graphics Meeting 2019*, Sarrebruck, dimanche 2 juin, 10 h 50 — enregistrement officiel publié par LGM, https://www.youtube.com/watch?v=04cnoNMdZBw (28 min 57 s, mise en ligne le 17 juillet 2019) : la diapositive intitulée « do-ocracy » apparaît à 17 min 31 s. Le résumé du programme annonçait un historique du projet, puis une discussion des « différentes approches de l'organisation d'équipe » et des enseignements tirés de 10 ans de groupe changeant https://libregraphicsmeeting.org/2019/sunday-02-06/index.html. Texte de la diapositive : « if there is a rule, it's don't stop anyone from coding when they're having fun » (notre traduction dans le corps du texte). ↩ ↩ ↩

21. La *do-ocracy* est répertoriée comme modèle de gouvernance à part entière : ceux qui font le travail prennent les décisions, l'autorité se proportionnant au temps et à l'énergie investis dans un domaine donné. Sur ces gouvernances informelles, leur fonctionnement et leur instabilité, voir Kostakis, V., Niaros, V. & Giotitsas, C. (2015). « Production and Governance in Hackerspaces : A

Manifestation of Informalism ? ». *International Journal of Cultural Studies*, 18(5), p. 555-573 https://doi.org/10.1177/1367877913519310 . ↩

22. Schneider, N. (29 avril 2020). « How's That Open Source Governance Working for You ? », Media Economies Design Lab, université du Colorado à Boulder https://www.colorado.edu/lab/medlab/2020/04/29/hows-open-source-governance-working-you — « basic tools for real accountability — the elections, the juries, the term limits — are nowhere to be found ». L'auteur ne traite pas de la *do-ocracy* sous ce nom, mais des gouvernances informelles du libre en général, dont il rattache explicitement le défaut à la tyrannie de l'absence de structure. ↩

23. Même conférence ([20]), diapositive suivante, à 18 min 38 s, intitulée « responsibility » : « whether we like it or not : people depend on our software to get their work done » (notre traduction dans le corps du texte). Le contenu qui lui est donné est la promesse de compatibilité indéfinie des retouches passées — l'engagement que le fichier annexe d'une photographie, rouvert des années plus tard, reproduise le même résultat. Promesse tenue, et coûteuse : elle contraint l'architecture du logiciel, comme le montre le chapitre « Le code est un passif ». ↩

24. Même conférence ([20]), diapositive « lessons learned », à 20 min 28 s : « gained what from laissez-faire ? 1. quick development, 2. features I never thought of, 3. translations I never knew existed, 4. technical debt :/ » (notre traduction dans le corps du texte). Le terme *laissez-faire* est employé

par les orateurs eux-mêmes pour désigner leur modèle d'organisation. ↩

25. Gunawardena, S. D. *et al.* (2022). « Destructive Criticism in Software Code Review Impacts Inclusion ». *Proceedings of the ACM on Human-Computer Interaction*, 6(CSCW2), art. 292 https://doi.org/10.1145/3555183 — la critique destructrice démotive tous les répondants, mais accroît disproportionnellement l'intention de sortie des développeuses et personnes non binaires. ↩

26. Terrell, J. *et al.* (2017). « Gender differences and bias in open source : pull request acceptance of women versus men ». *PeerJ Computer Science*, 3:e111 https://doi.org/10.7717/peerj-cs.111 — formulation prudente : taux d'acceptation des contributions des femmes supérieur dans l'ensemble, mais passant sous celui des hommes quand le genre devient identifiable. ↩

27. Miller, C. *et al.* (2022). « "Did You Miss My Comment or What ?" Understanding Toxicity in Open Source Discussions ». *Proceedings of ICSE 2022* (Distinguished Paper) https://doi.org/10.1145/3510003.3510111 — sur 100 fils toxiques GitHub, le motif dominant est l'utilisateur exigeant contre le mainteneur. ↩

28. Fil « Does the world need another incomplete photo editor ? », discuss.pixls.us, 6 mars 2016 https://discuss.pixls.us/t/does-the-world-need-another-incomplete-photo-editor/975 : la demande — « improve the stability and speed of existing software, without introducing

yet more “options”… Make the existing software more accessible » ; les réponses — un développeur du cœur : « Freeloaders are a burden and not an asset » ; un modérateur : « most of which are based on ignorance » (nos traductions dans le corps). ↩

29. Inventaire des fermetures de fils par l’API publique du forum, juillet 2026. La catégorie Darktable du forum principal ne compte qu’une vingtaine de fils fermés en 10 ans : hors doublons techniques, les fils de critique d’interface ou de direction y sont massivement sur-représentés. Exemples, avec le verbatim de fermeture : « Would a configuration wizard help new users of Darktable ? » (octobre 2025, fermé en deux jours — « Well, that’s enough I think ») ; « Color balance RGB chroma slider value range [POLL] » (décembre 2022, deux jours — « people will either take the survey or they won’t ») ; « Devs - How Hard Would It Be To… ? » (janvier 2023, 7 jours) ; « Another dt UI discussion » (juin 2026, 246 messages, 4 jours — « this has run its course »). Sur 16 fermetures après échauffement recensées de 2016 à 2026, 10 des 13 documentées jusqu’au dernier message sont closes par le même modérateur, avec un vocabulaire récurrent (« run its course », « that’s enough »), généralement deux à 9 jours après l’ouverture. Ces fils étaient réellement tendus au moment de leur fermeture — mais la tension naît précisément là où la critique d’interface rencontre la défense du projet, et la fermeture règle la tension, pas la critique. ↩

30. Suspension de deux mois : fil « Is development really important ? », août 2021 (« Enough is Enough », avec le rappel que le forum n'est « pas une zone de libre parole »). La formule « nobody is forcing you » est attestée de janvier 2018 à 2026, cinq fois sur sept sous la même plume de modérateur. ↩

31. PR #10694 (février 2022, ≈600 lignes) contre PR #11066 https://github.com/Darktable-org/Darktable/pull/11066 (février-avril 2022, +7 631 lignes) : le travail antérieur écrasé, et la question du doublon, posée par écrit, restée sans aucune réponse dans le registre. ↩

32. Chronologie sur le traqueur du projet, avril-mai 2022. PR #11066 https://github.com/Darktable-org/Darktable/pull/11066 fusionnée le 17 avril (8 participants), le module « recently used collections » supprimé le même jour. L'opposition : une douzaine de tickets en trois semaines, dont #11757 https://github.com/Darktable-org/Darktable/issues/11757 (ouvert le 7 mai par le mainteneur de la documentation ; 89 commentaires, 14 participants ; encore ouvert en 2026) — le développeur du cœur : « Current way is definitely not readable or intuitive » ; le responsable de la communauté : « my desire would be to revert the GUI pieces to what they are in the current 3.8.1 ». La révocation : PR #11752 https://github.com/Darktable-org/Darktable/pull/11752 , vote 6 pour / 4 contre, refusée par le mainteneur : « Let me even put this straight… I'm quite opposed to this PR » ; puis, dans #11757 : « Well I cannot force you to be happy :) » et « I'm

always in favor of moving forward ». Les motifs, dans #11747 https://github.com/Darktable-org/Darktable/issues/11747 (fermé le jour de son ouverture, étiqueté « controversial ») : « I started this huge work months ago… And now that it has land into master, at less than 1 week from freeze… no, we can't go back, I'm sorry » (l'auteur de la refonte) ; « To be fair most people don't like changing or moving forward… That's remind me all the haters about the new GNOME 3.0 » (le mainteneur) ; « I know kids scream from what they want until they have it, not adults » (#11757) ; « but stay constructive ». La concession : PR #11801 https://github.com/Darktable-org/Darktable/pull/11801 , fusionnée en 77 minutes le 14 mai ; les notes de version 4.0 https://github.com/Darktable-org/Darktable/releases/tag/release-4.0.0 annoncent la refonte sans mentionner la préférence de repli. Le ticket du mainteneur de la documentation fut fermé le 25 mai sur une infobulle, contre son avis ; il le rouvrit le 6 décembre. ↩

33. Export Element du salon Matrix public `#Darktable` (pont du canal IRC), 23 juillet 2026, 118 996 messages de février 2022 à juillet 2026 — en ma possession, vérifiable en rejoignant le salon. Le 7 mai 2022 : « WHAT ? » (12 h 57), « oh fucking shit, progress got in the way of productivity once again » (12 h 58), lien vers le rapport de bug (13 h 26). Le même journal établit mon propre registre d'août 2022 — volume au plus haut en mai (2 880 messages), invectives en août — et l'échange du 8 août analysé au chapitre « Écouter sans obéir » : la pièce charge les deux camps, et c'est à ce titre que je la verse au dossier. ↩ ↩ ↩ ↩ ↩ ↩ ↩ ↩ ↩ ↩

34. Ticket #12319 https://github.com/Darktable-org/Darktable/issues/12319, « Define and implement a process for large changes to the project », ouvert le 13 août 2022 par le responsable de la communauté (7 approbations), citant nommément « the collection filter changes » parmi les cas à encadrer ; fermé le 18 août : « Alright. We're already doing the best we can then. » ↩

35. #11757 https://github.com/Darktable-org/Darktable/issues/11757, 6 décembre 2022, un développeur du cœur : « Imho the biggest mistake was to not keep the old one as default » — et le principal défenseur de la refonte : « you're probably right… keeping old widget by default would have been better ». Même fil, 30 juillet 2026 : « Then the originator and me got shut up :-) ». ↩

36. Hirschman, A. O. (1970). *Exit, Voice, and Loyalty*. Cambridge, MA : Harvard University Press (trad. fr. *Défection et prise de parole*, Fayard, 1995) https://openlibrary.org/works/OL2745855W — quand la prise de parole est rendue inopérante, il reste la défection ; le fork est la forme pure de la défection logicielle. ↩

37. Journal du salon Matrix ([33]), verbatims (notre traduction dans le corps du texte) : mainteneur de la documentation, 7 mai 2022 — « it felt like a dev scratching an itch » — et 25 mai 2022 — « wow Pascal thought the tooltip was *the* fix to the readability problems and closed my issue » ; semaine du départ : « Thanks for the work and the rant ! » (27 juin), « It looks like a bit of a worry » (29 juin, sur la nouvelle interface

de la branche principale). En public, sur le forum : « AP chose to use those words as is his right » (https://discuss.pixls.us/t/31388/31 , 28 juin). Les soutiens du salon, eux, ne furent jamais relayés sur le dépôt ou le forum public. ↩

38. La vidéo : publiée le 27 juin 2022, discutée le jour même sur https://discuss.pixls.us/t/31388 (ouverture par l'utilisateur vbs : « the language was a bit too strong … I think there are a lot of good points in the video » ; archivé : https://web.archive.org/web/20260723092719/https://discuss.pixls.us/t/r-Darktable-options-and-path-forward-and-user-point-of-view/31388) ; son lien fut retiré du fil par la modération pour « very strong language ». L'article : « Darktable : dans le mur au ralenti », https://ansel.photos/fr/news/Darktable-dans-le-mur-au-ralenti/ , 15 février 2023 (date d'en-tête du site ; archivé : https://web.archive.org/web/20260720234454/https://ansel.photos/fr/news/Darktable-dans-le-mur-au-ralenti/) — environ 10 000 mots, sources et exemples à l'appui. L'absence de réponse de fond est documentée quantitativement plus loin dans ce chapitre (l'analyse des 3 niveaux). ↩

39. Même locuteur — paperdigits, administrateur du forum —, discuss.pixls.us : 2022-07-04, « Nobody has been rejected » (https://discuss.pixls.us/t/31388/153 ; archivé : https://web.archive.org/web/20260723092318/https://discuss.pixls.us/t/r-Darktable-options-and-path-forward-and-user-point-of-view/31388/153) et 2022-06-29, « he asked to be removed and I obliged him »

(https://discuss.pixls.us/t/31388/54 ; archivé : https://web.archive.org/web/20260723094014/https://discuss.pixls.us/t/r-Darktable-options-and-path-forward-and-user-point-of-view/31388/54) ; 2026-06-20, « it is our fault (particularly mine) for tolerating such poor behavior for so long » (https://discuss.pixls.us/t/58735/10 ; archivé : https://web.archive.org/web/20260721230927/https://discuss.pixls.us/t/aps-new-video-on-ansel/58735/10) ; « thrived since he left » : https://discuss.pixls.us/t/58735/49 (archivé : https://web.archive.org/web/20260723092430/https://discuss.pixls.us/t/aps-new-video-on-ansel/58735/49). Le journal Matrix ([33]) date l'écart entre les deux registres du même locuteur à 3 jours près : le 1er juillet 2022, il m'écrit en salon « that's why you'll end up by yourself » et qualifie ma vidéo de « fucking pathetic and sad » ; le 4 juillet, sur le forum public : « Nobody has been rejected ». Le registre des salons est plus cru que celui des forums pour tout le monde, moi compris ; ce que la paire établit n'est pas un trait de caractère, c'est l'écart entre la scène et les coulisses au moment même où le récit officiel se formait. ↩

40. Festinger, L. (1957). *A Theory of Cognitive Dissonance*. Evanston : Row, Peterson (trad. fr. *Une théorie de la dissonance cognitive*, Enrick B., 2017) https://doi.org/10.1515/9781503620766 ; Halbwachs, M. (1950). *La mémoire collective*. Paris : PUF (éd. critique Albin Michel, 1997) https://openlibrary.org/works/OL1232451W — la mémoire du groupe se reconstruit au service de son présent. ↩

41. Verbatim, discuss.pixls.us, 2022-07-03 : « Mostly I'm hearing the equivalent to good riddance ! I find it astonishing that the person who has brought Darktable to a completely new level in terms of color processing and scene referred workflow feels like he has to leave the project » (https://discuss.pixls.us/t/31388/150, utilisateur anon16817310 ; archivé : https://web.archive.org/web/20260722032615/https://discuss.pixls.us/t/r-Darktable-options-and-path-forward-and-user-point-of-view/31388/150). Message d'un utilisateur tiers, contemporain du départ, contredisant en temps réel le « personne n'a été rejeté » officiel de la même semaine. ↩

42. Verbatim, discuss.pixls.us, 2023-03-26 : « This has all the look and feel of a mob attack, one where the participants are intoxicated breathing their own exhaust » (https://discuss.pixls.us/t/33910/198, utilisateur meaningfullyhappy ; archivé : https://web.archive.org/web/20260722032643/https://discuss.pixls.us/t/ansel-was-r-Darktable/33910/198). Message d'un tiers décrivant le fil en cours. ↩

43. Leymann, H. (1990). « Mobbing and Psychological Terror at Workplaces ». *Violence and Victims*, 5(2), 119-126 https://doi.org/10.1891/0886-6708.5.2.119 ; et (1996) « The Content and Development of Mobbing at Work », *European Journal of Work and Organizational Psychology*, 5(2), 165-184 https://doi.org/10.1080/13594329608414853. Version française : Hirigoyen, M.-F. (1998). *Le harcèlement moral : la*

violence perverse au quotidien. Paris : Syros. https://openlibrary.org/works/OL572454W ↩

44. Contributeurs : chapitre « Livrer, pas larguer », historique git dédupliqué. Salon : journal Matrix ([33]) — messages des autres membres, moyenne mensuelle de mars-août 2022 (ma présence) : 6 786 ; d'octobre 2022 à mars 2023 : 3 174 (− 53 %) ; fin 2023 : environ 1 200 (− 82 %). Entraide : réponses adressées (citations et interpellations nominales, méthode du chapitre « Le travailleur invisible ») — 4 905 en 2022, 1 585 en 2023, environ 1 500 par an depuis. Le creux n'est jamais suivi d'un rebond, ce que « prospérer » exigerait. Borne du test : il porte sur les lieux propres du projet — son salon, son dépôt, son entraide —, pas sur tout l'écosystème. Le forum francophone tiers (Darktable.fr, 5 101 fils comptés par l'API), en déclin depuis son sommet de 2018 pour des causes antérieures au conflit — la migration du public vers le forum anglophone —, montre au contraire un rebond en 2026. Corrigé du bruit des fils de présentation que le forum impose aux nouveaux inscrits : environ 245 fils de contenu en rythme annuel contre 150 en 2025 (+ 63 %), et les présentations elles-mêmes vont à près de 4 fois le rythme de 2025 — un afflux de nouveaux membres. Je le signale parce que le test doit rapporter aussi ce qui remonte. ↩

45. « Worried about LLM-written modules » https://discuss.pixls.us/t/worried-about-llm-written-modules/56430, 4 mars 2026, 265 messages — l'ouverture, par un contributeur du projet : « Within a week, we've seen 2 new module proposals, both created by people who made

it clear… that they had an idea, and used an LLM to quickly come up with an implementation » (notre traduction dans le corps) ; « On module proliferation » https://discuss.pixls.us/t/on-module-proliferation/56435 , 5 mars 2026, 102 messages. Les jouets nommés dans le corps ont leurs fils propres, les mêmes semaines : « Contrast Management RGB : A new scene-referred approach (POC) », 359 messages ; « [New module] Color harmonizer », 228 messages. ↩

46. Décomposition par genre des fils : mots-clés de titre, puis revue manuelle des 4 plus gros trimestres et des 25 plus gros fils, 48 fils reclassés un à un — dont un second passage d'audit sur les trimestres sommets de 2023 et 2024, qui n'a déplacé que 3 pour cent de leurs messages : ces sommets-là sont bien de l'entraide — listes, règles et identifiants en corpus. Parmi les reclassés : « Editing moments » (1 322 messages, vitrine de photographies), « current OSX Build » (1 004, coordination de compilation), « Should Darktable remain AI-free ? » (724, gouvernance), « New Sigmoid Scene to Display mapping » (808, développement de module). L'entraide nettoyée, mes messages exclus : 4 155 messages en 2019, 7 937, 8 455, 10 114, puis le sommet en 2023 (10 491), 9 119, 8 985, environ 9 700 en rythme 2026 — une croissance jusqu'en 2023, portée notamment par les questions de migration depuis Lightroom, puis une érosion lente. Les délibérations de conception : entre 21 et 340 messages par an de 2020 à 2024, puis 1 893 en 2025 et un rythme de plus de 3 200 en 2026. Les annonces et préversions : 3 à 6 % des messages avant 2025, un tiers en

2026 ; « On module proliferation », 2026, documente la lassitude du forum envers ses propres faiseurs. ↩

47. Avril 2023, forum communautaire : « it's not worth to discuss about people that are not present here » ; « if you want to talk about Ansel with its developer, you cannot do so on pixls ». Novembre 2025 : « I would be in favour of the moderators applying a version of Godwin's law whenever someone tries to bring this whole thing up again ». « Travailler en groupe » : juillet 2022, en réponse à une demande de réponse point par point — « this guy his really good at making nice modules but cannot work in a group of people » (sic). ↩ ↩

48. Verbatim, discuss.pixls.us, 2023-04-02 : « The technical merits of his code, modules, vision and whatever else pale in comparison to his toxic behavior » (https://discuss.pixls.us/t/36370/2 ; archivé : https://web.archive.org/web/20260723092528/https://discuss.pixls.us/t/we-have-a-problem-what-is-it-about-ansel-and-ap-edit-concern-fully-resolved-via-respectful-dialogue/36370/2) ; et Darktable-org/Darktable, PR #12233, 2022-08-21 : « until Aurélien learn to speak to others I won't be given a single though to what he has to say, period » (https://github.com/Darktable-org/Darktable/pull/12233 ; archivé : https://web.archive.org/web/20260722015957/https://github.com/Darktable-org/Darktable/pull/12233). L'analyse quantitative des réponses — faits d'implémentation parfois discutés, mesures de maintenabilité jamais (zéro occurrence

de « cyclomatic » ou « sonarcloud » dans l'histoire du forum), conception réglée par l'autorité — porte sur 25 affirmations sourcées confrontées à plus de 2 000 messages. ↩

49. Registre publié 2022-2023 : volumes de code, requêtes SQL comptées, scores SonarCloud — par exemple, complexité cyclomatique de la bibliothèque de contrôles d'interface : 735, dette technique estimée à 1 jour 7 heures ; après réécriture dans Ansel : 494 (https://ansel.photos/en/news/unDarktable-ing-gui-controls/). Sur tout le registre du forum communautaire, les termes « cyclomatic », « cognitive complexity » et « sonarcloud » n'apparaissent pas une seule fois : les chiffres n'y ont jamais été discutés, ni repris, ni contestés. ↩ ↩

50. Verbatim, discuss.pixls.us, 2023-03-20 (https://discuss.pixls.us/t/33910/85 , utilisateur LebedevRI) : « The best Darktable i ever used was Darktable-1.4. Every release, every day since, it has been actively and gradually decaying. » (archivé : https://web.archive.org/web/20260723092117/https://discuss.pixls.us/t/ansel-was-r-Darktable/33910/85). Même message que l'endossement des 60-80 % cité plus bas. ↩

51. Darktable-org/Darktable, issue #11747, 2022-05-07 — réponse de l'auteur de la refonte (AlicVB) : « I started this huge work months ago. everything has been transparent… And now that it has land into master, at less than 1 week from freeze… no, we can't go back, I'm sorry. » Le travail investi et le calendrier y tiennent lieu d'argument ; la validité de la critique n'y est pas discutée.

(https://github.com/Darktable-org/Darktable/issues/11747 ; archivé : https://web.archive.org/web/20260723092220/https://github.com/Darktable-org/Darktable/issues/11747) ↩

52. La controverse : PR #20322 (« AI inference subsystem », février-mars 2026) et fil « Should Darktable remain AI-free ? » https://discuss.pixls.us/t/52778 (724 messages, clos). Le responsable de la communauté : « Trying to speak up an make my opinion heard resulted in the maintainer telling me I was threatening the project » et « I've left because of your treatment and actions directly » (messages #516 et #530 https://discuss.pixls.us/t/52778/516) ; les trois griefs, dont le vote de « core devs » excluant les autres mainteneurs : #556 https://discuss.pixls.us/t/52778/556 ; le silence : #600 https://discuss.pixls.us/t/52778/600 ; sur GitHub, après le vote : « When a project let's you know that your contributions are worthless, listen to them » (PR #20322, 18 février 2026). Le mainteneur, en face : « He told me clearly that if AI was introduced in Darktable he will quit the project. That's just plain threatening » (#529 https://discuss.pixls.us/t/52778/529) ; et « You've expressed your opinion (and elstoc also)… The doc maintainers were against AI fine. Now I wanted to know what the devs thought about this » (#557 https://discuss.pixls.us/t/52778/557). Le retrait du mainteneur de la documentation : « ultimately felt it was better for the project for me to stand aside » (fil « New dtdocs maintainer(s) needed », message #3 https://discuss.pixls.us/t/57489/3) ; ses fusions sur le dépôt

de documentation s'arrêtent le 9 février 2026 — la PR litigieuse ouvre le 11 — et ne reprennent, par d'autres mains, que le 10 mai. L'annonce, par le mainteneur : « Mica & Elstoc are no more interested in maintaining the Darktable documentation » (#1 https://discuss.pixls.us/t/57489/1) ; la réponse du responsable de la communauté : « it was your combination of action and inaction… from your position of power as maintainer of the project that lead to me leaving the project… But adding insult to injury just seems to be how you do things » (#2 https://discuss.pixls.us/t/57489/2). Nos traductions dans le corps. ↩ ↩ ↩

53. Noelle-Neumann, E. (1974). « The Spiral of Silence : A Theory of Public Opinion ». *Journal of Communication*, 24(2), 43-51 https://doi.org/10.1111/j.1460-2466.1974.tb00367.x ; Kuran, T. (1995). *Private Truths, Public Lies : The Social Consequences of Preference Falsification*. Cambridge, MA : Harvard University Press https://openlibrary.org/works/OL3527602W . Le témoignage IRC relève du régime du chapitre « Écouter sans obéir » (canal non archivé) ; la citation du mainteneur (« people who like it do not voice it so loudly ») est, elle, publique : https://discuss.pixls.us/t/58799/228 (archivé : https://web.archive.org/web/20260723093046/https://discuss.pixls.us/t/another-dt-ui-discussion/58799/228). ↩

54. L'argument : « Darktable is plug-in based… every module is self contained and does not know about other plug-ins. So it is harder to have specific control order on screen » (discuss.pixls.us, fil « Another dt UI discussion », message 22,

juin 2026 https://discuss.pixls.us/t/58799/22 ; notre traduction). Audit du code mené le 22 juillet 2026 sur le dépôt local, version antérieure au fork (24 mai 2022) comparée à la version courante d'Ansel, mêmes métriques. Darktable : 718 références à des noms de modules hors du répertoire des modules, réparties dans 31 fichiers ; `src/common/iop_order.c` (2 211 lignes) contient à lui seul 3 ordonnancements complets et codés en dur de l'ensemble des modules — l'ordre ne « résulte » pas de greffons qui s'ignorent, il est une table centrale que le cœur consulte en permanence. Le monolithe subsiste dans Ansel, aux mêmes critères, avec une amplitude moindre (hardcodage des groupes de modules ramené de 245 entrées à 0, contact avec la structure globale de 98 % des fichiers à 93 %). ↩

55. Salon Matrix du projet (export privé, cf. [33]), le mainteneur de la documentation : 20 mars 2023, « this made the ui look cluttered » ; 23 mars 2023, « I can't stomach the endless arguments with pascal, which often end up with worse UX as he tries to satisfy everyone » ; 13 janvier 2024, « current interface is just bad ». En public, le mainteneur, 26 juin 2026 (https://discuss.pixls.us/t/58799/228 et 230 https://discuss.pixls.us/t/58799/230) : « I do like a lot the way the UI is… The Darktable devs like the way the UI is. » ↩

56. discuss.pixls.us, fil 33910, messages 198 et 202 https://discuss.pixls.us/t/33910/198, 26 mars 2023 : « This has all the look and feel of a mob attack » ; « No one person can create a "toxic atmosphere." It requires the participation and complicity of several… if that "atmosphere is still here" in

the absence of the person being blamed for “creating it”, it confirms… the complicity and participation of those others » (notre traduction dans le corps). ↩

57. Inventaire par l’API du forum : 33 fils postérieurs au départ instruisant mon cas hors ma présence. La loi de Godwin : cf. [47]. L’image, 10 juillet 2026 : « People not knowing well Darktable and Ansel could think when reading AP post that we are a bunch of stupid people making a crap software named Darktable. This is just plain wrong. » ↩

58. Reprises avec attribution : Darktable-org/Darktable, PR #20811 https://github.com/Darktable-org/Darktable/pull/20811 et #20798 https://github.com/Darktable-org/Darktable/pull/20798 (avril 2026), rétroportages de correctifs Ansel au commit près — une issue que les initiés anticipaient dès janvier 2023, sur le salon Matrix du projet : « nous devrons peut-être les porter depuis Ansel » (9 janvier), « je pense que pratiquement tous les commits d’AP seront récupérés depuis Ansel » (17 mars ; notre traduction — journal cité en [33]). Endossement : « i fully and wholeheartedly agree with 60 %…80 % of what is said in that post » (https://discuss.pixls.us/t/33910/85 , 2023-03-20, archivé : https://web.archive.org/web/20260723092117/https://discuss.pixls.us/t/ansel-was-r-Darktable/33910/85 ; contributeur de 10 ans — aujourd’hui mainteneur d’une bibliothèque tierce et lui-même en désaccord avec la gouvernance actuelle : corroboration d’un critique aligné, non verdict interne). Estimation GTK 4 refaite en 2026 (≈ 40 000 lignes, 18-24

mois) concordant avec mes chiffres de 2023, sans référence. ↩ ↩

59. Comptage par script sur l'API publique du forum, 23 juillet 2026 : 14 fils (2019-2026) portant sur mon travail ou mon départ, 1 931 messages, citations imbriquées exclues pour ne pas compter deux fois le même texte, et mes propres messages (102 sur 1 931) exclus du comptage. Avant le 27 juin 2022 : 65 désignations nominales (« Aurélien », « Aurelien », ou mention du compte), 0 « AP ». Du 27 juin 2022 au 30 juin 2023 : 75 nominales, 142 « AP » — 65 %. En 2026, la part se maintient au-delà de la moitié ; le semestre en cours (partiel) ne contient plus que des initiales. Première occurrence : https://discuss.pixls.us/t/31388/3, 28 juin 2022. Biais connu, défavorable à la mesure : l'anonymisation de mon compte — à ma propre demande, au titre du RGPD — a réécrit rétroactivement les mentions @aurelienpierre en @anon41087856, comptées ici comme nominales quelle que soit leur date. L'économie de frappe n'explique pas la bascule : l'abréviation était disponible pendant 3 ans et n'a pas servi une seule fois avant le conflit. Les formes sans nom (« a previous developer », « that person is no longer here » : https://discuss.pixls.us/t/54423/138, 2026-01-06 ; « the person who designed » : https://discuss.pixls.us/t/58799/243, 2026-06-26) sont trop rares pour une statistique et valent comme jalons qualitatifs. Réplication indépendante : le journal du salon Matrix du projet ([33]) montre le même basculement — tant que j'y suis, le prénom domine largement (623 mentions contre 34 au deuxième trimestre 2022) ; le trimestre où je quitte le salon, la bascule (8 contre 48 au 4e trimestre), puis

le sigle règne pour de bon. Deux lieux, deux corpus, le même pli au même moment. ↩

60. Merton, R. K. (1968). *Social Theory and Social Structure*, éd. augmentée. New York : Free Press, p. 25-38 https://openlibrary.org/works/OL269872W ; définition verbatim dans Merton, R. K. (1988). « The Matthew Effect in Science, II ». *Isis*, 79(4), p. 622 https://doi.org/10.1086/354848 : « the obliteration of the sources of ideas, methods, or findings by their being anonymously incorporated in current canonical knowledge » (notre traduction dans le corps du texte ; le français « oblitération par incorporation » est attesté dans la littérature mertonienne francophone). Merton décrit un processus normal de la science, lent et sans hostilité ; l'appliquer à un cas où l'effacement du nom est contemporain d'un conflit est mon extension, et je la signale comme telle. Les réutilisations du tableau sont sourcées aux notes [58] et [33]. ↩

28. Le conflit bien tempéré

Après le chapitre qui précède, la tentation est connue d'avance : moins de conflit. Des communautés apaisées, des désaccords feutrés, de la bienveillance en toutes choses. Ça n'est pas juste terriblement naïf, c'est un contresens, à lever. Le cercle fermé n'était pas malade d'un excès de conflit ; il était malade de conflits *sans règles* — tranchés par les moyens, jamais sur le fond — et, tout autant, de conflits *empêchés* : la spirale du silence qui privait le désaccord de voix publique était une machine à pacifier, et elle a produit le pire. L'absence de dispute n'est pas la paix : c'est souvent le signe que quelqu'un a déjà gagné sans avoir eu à débattre. Ce chapitre écrit les règles qui ont manqué — pour tout projet qui préfère apprendre de ses conflits plutôt que d'y brûler.

Les gens qui tiennent à quelque chose se battent pour elle : le conflit naît quand des personnes ont des intérêts inconciliables, jugent l'affaire assez importante pour s'y engager, et ont les ressources de le faire. C'est un processus sain : on y affirme sa volonté, on y pèse sur son propre avenir. Le refuser parce qu'il serait « violent », c'est se soumettre d'avance au parti le plus puissant : celui qui gagne sans se battre, par simple inertie, en attendant que les autres s'épuisent. Or le chapitre « La violence à trois étages » a défini la violence par ce qu'elle produit, la souffrance, et non par le bruit qu'elle fait. Un statu quo qui broie en silence est violent. Le combat bruyant qui le conteste ne l'est pas nécessairement. Une communauté sans aucune dispute

visible devrait inquiéter, pas rassurer : ou bien plus personne n'y tient à rien, ou bien ceux qui y tiennent ont déjà été réduits au silence — et le chapitre précédent a montré lequel des deux c'était.

La recherche sur les organisations confirme cette réhabilitation, et la précise d'une distinction capitale : il y a conflit et conflit. Le conflit *de tâche* — sur le fond : ce problème, cette solution, ces critères — améliore les décisions ; la controverse organisée, où des positions opposées s'instruisent mutuellement, produit de meilleurs jugements que l'unanimité spontanée[1]. Le conflit *de personnes* — qui vise l'autre, sa valeur, ses intentions — dégrade tout ce qu'il touche. Et le résultat le plus utile de cette littérature est leur relation : le conflit de tâche ne porte ses fruits que là où le conflit de personnes est contenu, c'est-à-dire là où l'on peut attaquer une idée sans que son auteur se sente attaqué — la sécurité psychologique du chapitre « Le comité n'inventera rien », encore elle[2]. Tout l'art est donc de tenir les deux gestes à la fois : chauffer le débat sur les objets, le refroidir sur les personnes. Le chapitre « Le cercle fermé » a montré le double échec symétrique : un milieu qui tolérait l'attaque des personnes, et refusait le débat de fond.

Or ce livre a passé sa première moitié à construire, sans le dire, une machine à convertir les conflits de personnes en conflits de tâche. Reprenez l'appareil dans cet éclairage. Le problème posé par écrit (chapitre « Poser le problème ») : tant que le problème est implicite, s'y opposer, c'est s'opposer à quelqu'un ; écrit, il devient un objet que l'on peut attaquer sans toucher personne. Les critères de validation posés avant la solution : ils déplacent la

dispute de « ton idée contre la mienne » vers « laquelle satisfait les critères » — et la clôture (chapitre « Décider, puis tenir ») donne au conflit une fin possible, ce que le chapitre « Le mot et la chose » appelait la règle d'arrêt. La mesure (chapitres « Mesurer sans se mentir » et « Compter les absents ») : un chiffre auditable est un tiers impartial en miniature, le seul témoin que les deux camps peuvent interroger. La remontée aux besoins (chapitres « La racine et le symptôme » et « Poser le problème ») : la plupart des conflits de solutions sont des conflits de positions, et se dissolvent quand on remonte aux besoins — le dîner et le cinéma du chapitre « Poser le problème ». Le témoignage contre la prescription (chapitre « Écouter sans obéir ») : il désarme le conflit d'autorité en donnant à chacun un domaine où il est souverain et un domaine où il ne l'est pas. Rien de tout cela n'élimine le conflit : tout cela le *civilise* — au sens propre, le fait passer du duel au procès.

Le même appareil, lu en négatif, explique enfin pourquoi les conflits du logiciel libre sont si uniformément *vécus* comme des conflits de personnes — et pourquoi ils le deviennent. Dans un projet sans cahier des charges, un désaccord de conception n'a littéralement aucun objet où s'accrocher. Aucun but écrit : s'opposer à une proposition, c'est s'opposer à son auteur. Aucune priorité arbitrée : chaque camp invoque la sienne. Aucune définition de ce qu'« optimal » veut dire : optimal en vitesse contre optimal en lisibilité, optimal pour mon flux de travail contre optimal pour le tien, sans qu'aucun texte ne dise lequel le projet a promis. Le désaccord, privé d'objet, se rabat alors sur le seul objet disponible : les personnes. Ce n'est pas que les

caractères y soient pires qu'ailleurs ; c'est que le déficit de méthode technique se convertit mécaniquement en dommage relationnel — le miroir exact du chapitre « Il n'y a pas de problème technique » : il n'y a pas de problème technique, mais une méthode technique absente fabrique des problèmes humains. Et la boucle s'auto-alimente : une fois les relations abîmées, proposer d'écrire enfin les buts et les critères ressemble à une manœuvre du camp adverse — la méthode exige précisément la confiance que son absence a détruite. Un conflit sans critère de résolution ne se résout pas ; il se *termine*, et il ne peut se terminer que par la domination : l'ancienneté qu'on fait valoir, le droit de fusion, le veto — le chapitre « Le cercle fermé » a documenté ce régime. Quant aux utilisateurs, leur place dans ce théâtre dit tout : ils mangent du pop-corn. Les fils de discussion en témoignent — des dizaines de spectateurs suivent la controverse comme un feuilleton, prennent parti pour des personnages, jamais pour des critères, et pour cause : il n'existe aucun critère qu'ils pourraient invoquer. Dans un projet doté d'un cahier des charges, ils seraient les arbitres naturels — leurs besoins, consignés, départageraient les camps ; c'est même l'une des fonctions d'un cahier des charges que de donner au conflit un juge qui n'est aucune des parties. Sans lui, le siège de l'arbitre est vide, et c'est ce siège vide que le pouvoir de fait vient occuper.

Le siège vide n'appelle pas que le pouvoir de fait ; il appelle une norme de substitution. Faute d'un critère qui trie le vrai du faux, une communauté ne peut policer que ce que tout le monde sait voir : le ton. La justesse d'une affirmation demande une expertise et un arbitrage qu'elle n'a pas ; la grossièreté d'une réponse,

n'importe qui la constate. Elle sanctionne donc la seule chose qu'elle sait sanctionner, et laisse passer l'autre : l'erreur assurée, pourvu qu'elle soit polie. Ce partage n'a rien de propre au logiciel. Les disciplines dont l'erreur tue — la médecine, le génie civil — se sont dotées de codes, de normes et d'un devoir de diligence, précisément parce qu'elle tue ; celles dont l'erreur ne tue personne, comme le traitement d'images, n'ont rien de tel, et l'absence de sanction s'y lit comme un permis de négligence[3]. Mais l'erreur qui ne tue pas coûte quand même : elle égare, elle désinforme, elle fait perdre à tous un temps qui ne revient pas — l'utilisateur devra désapprendre ce qu'il croyait savoir, et celui qui conçoit devra désamorcer la fausseté avant même de pouvoir expliquer son outil. Un désaccord de fait, sans métrique du vrai, ne peut donc pas se clore sur les faits : il se rabat sur le ton, où la seule chose punissable est le registre. Le ton devient le juge que la communauté n'a pas — et un juge pervers, qui protège l'erreur polie et condamne la justesse abrupte.

Avant de dessiner le procès, il faut régler un cas que le far-west produit en série, et qui piège une chose vraie. Concevoir en source ouverte, c'est concevoir sous un regard public permanent : chaque détail sera critiqué, toujours, par vagues, à contretemps, souvent par des gens qui n'ont pas lu le fil et ne le liront pas. Ce feu roulant épuise, et le concepteur a un besoin légitime de s'en abriter — ce livre a assez mesuré la fatigue pour ne pas le contester. Mais le bouclier légitime dégénère en immunité, et j'en donne un cas daté, où je suis partie. En mai 2022, je signalai par un rapport de bug les régressions de la refonte des filtres de collection ; l'auteur de la refonte clôtura sa réponse ainsi : « je

suis désolé d'être aussi rude, mais j'affronte quotidiennement, dans mon vrai travail, des remarques et des injonctions du type "c'était mieux avant"… et c'est simplement absurde… le fait de gens qui (à dessein ou non) ignorent la réalité et la façon dont le monde réel a changé »[4]. La réponse est fallacieuse deux fois, et il faut prendre le temps de l'établir, car c'est le mécanisme générique qui importe. Quelle réalité avait changé ? Pas la photographie — ni comme métier, ni comme pratique ; rien, dans le monde réel, n'avait requis cette refonte. La seule chose qui avait changé était l'outil, et il avait changé parce que son auteur l'avait changé : invoquer « le monde réel » pour couvrir sa propre décision, c'est faire passer un choix de conception pour une loi de la nature — et l'on notera que l'épuisement invoqué vient d'ailleurs, de son emploi, importé dans l'échange pour disqualifier un interlocuteur qui n'y était pour rien. Seconde fallacie : la critique ne venait pas d'un anonyme de passage, mais d'un développeur du cœur du projet, coéquipier de longue date sur le même dépôt. Et fût-elle venue d'un anonyme, sa substance n'en aurait pas pesé moins — car elle se mesure. L'ancienne fonction de filtrage : 548 lignes, 22 fonctions, complexité cyclomatique moyenne de 1,9, aucune fonction au-dessus du plafond que la métrique recommande. Le système qui l'a remplacée : 5 207 lignes réparties en 16 fichiers, 147 fonctions, complexité moyenne de 4,7, dont deux fonctions au-delà de ce plafond[5]. On peut discuter des goûts ; on ne discute pas d'un facteur 10 en volume de code et d'un facteur deux et demi en complexité, pour remplacer une chose qui existait, fonctionnait, et se laissait comprendre. Le bouclier contre le harcèlement servait ici à repousser un rapport de bug argumenté : c'est l'immunité par

l'épuisement — la fatigue, réelle, convertie en dispense de fond, permanente.

L'autre visage du même bouclier est le récit héroïque. J'ai lu, dans le salon Matrix francophone des développeurs, sous la plume du mainteneur qui fusionne l'essentiel du code depuis des années, l'aveu qu'il est parfois dur de prendre toutes les décisions seul, puis d'essuyer les reproches après coup — je cite de mémoire[6]. L'aveu est humain, et il appelle deux sorties connues : déléguer des responsabilités, ou consulter ceux que la décision affecte avant de la prendre. Le registre des fusions, au chapitre « Penser contre soi », montre qu'aucune des deux n'a été prise — la concentration s'est accrue d'année en année. Reste donc la troisième voie, celle du récit : la figure du chef nécessairement seul, qui porte le fardeau et endure l'ingratitude. Cette figure a une fonction précise : elle garde le pouvoir et esquive le devoir de rendre des comptes — la solitude y est renversée en légitimité au lieu d'être traitée comme le symptôme d'un défaut d'organisation. Et elle s'auto-alimente : parce qu'il est seul, il décide seul ; parce qu'il décide seul, les reproches pleuvent après coup ; parce qu'ils pleuvent, la solitude du chef est confirmée, et le cercle se referme en hagiographie. On n'en sort pas par la psychologie ; on en sort par la structure.

Ces deux figures — l'immunité par l'épuisement, la solitude par vocation — sont le même défaut vu de deux places. Le besoin de protection du concepteur est légitime ; le besoin d'expression de l'usager l'est aussi ; et le far-west les fait se détruire l'un l'autre. Sans canal institué, la critique de fond n'a qu'une porte d'entrée : la pression — l'insistance, la vague, le pilori — c'est-à-dire ce qui

confine au harcèlement ; le système encourage précisément ce dont il se plaint, parce que c'est la seule voie qui obtienne parfois une réponse. Et encore n'est-ce pas une voie : la défense est symétrique de l'attaque, chapitre « Le cercle fermé ». Un système sous tension permanente, sans décharge organisée, a besoin d'un fusible — faute de quoi il finit en église : des comportements où plus rien n'est rationnel, plus rien n'est falsifiable, et où l'on ne débat plus, on communie[7] — le chapitre « Le culte du cargo » a décrit ce régime, et le chapitre « Le mot et la chose » a dit pourquoi un design infalsifiable n'est plus un design. Le fusible porte un nom moins exaltant que la liberté : des institutions.

Reste ce que le procès exige et que le far-west n'a pas : des institutions, même minuscules. Elinor Ostrom, en étudiant les communs qui survivent aux siècles, en a fait l'un de ses principes : les communautés durables disposent toutes de mécanismes de résolution des conflits *rapides et peu coûteux* — pas de grands tribunaux, des arènes accessibles où les différends se règlent avant de s'envenimer[8]. Transposé à un projet, même petit, cela tient en 4 étages que l'on peut écrire sur une page. L'auto-règlement d'abord : la discussion structurée du chapitre « Le comité n'inventera rien » — un sujet, un lieu, les hors-sujets déportés, un délai. La médiation ensuite : un tiers accepté des deux camps, qui ne tranche pas mais force chacun à reformuler la position de l'autre jusqu'à ce qu'elle soit reconnue — la moitié des conflits meurent là, faute de combattants, quand chacun découvre ce que l'autre voulait vraiment dire. L'arbitrage après : quelqu'un, désigné d'avance et pour un temps, tranche sur le fond, par écrit, critères à l'appui — et sa décision s'applique,

révisable par les voies du chapitre « Décider, puis tenir », du nouveau ou rien. Le vote enfin, en dernier recours, pour les questions de cap qui relèvent du collectif — jamais pour le design, chapitre « Le comité n'inventera rien ». Quatre étages, une page ; l'essentiel n'est pas leur sophistication, c'est qu'ils existent *avant* le conflit. Un protocole écrit à froid protège tout le monde ; un protocole improvisé à chaud n'est qu'une arme de plus dans le far-west. Une formule entendue en chemin, dont je n'ai pas retrouvé l'auteur, résume tout : la démocratie, c'est se mettre d'accord sur des protocoles de désaccord. La science politique dit la même chose en langage de traité. La démocratie ne résout pas les conflits, elle les traite. Des règles purement procédurales suffisent à les structurer et à les contenir. Et c'est cela qui distingue la paix civile de la paix par la force[9].

À quoi s'ajoutent 3 règles d'hygiène que mon expérience impose d'écrire. Traiter les conflits *tôt* : le chapitre « Le cercle fermé » raconte des années de griefs accumulés en privé puis déchargés d'un coup ; un conflit traité à la semaine est une conversation, le même conflit traité à l'année est une guerre — et la dette de conflit court aux mêmes intérêts que la dette technique. Protéger le dissident, pas seulement la majorité : les contre-mesures du chapitre « Le comité n'inventera rien » — votes anonymes, avocat du diable, dissensus institué — existent précisément pour que le désaccord n'ait pas à choisir entre le silence et l'héroïsme ; une communauté où il faut du courage pour être en désaccord est déjà malade. Et appliquer les règles symétriquement : le dossier du chapitre précédent montre ce qu'il advient d'une modération qui pèse les mots du critique et pas ceux de ses adversaires — une

police perçue comme un camp cesse d'être une police, et chaque sanction asymétrique fabrique du ressentiment qui resurgira ailleurs, plus laid.

L'objection est économique, et elle est de bonne foi : tout cela est de la bureaucratie, un petit projet n'a pas les moyens de se payer des procédures, on verra quand on sera grands. Deux réponses. La première est comptable : le coût du conflit non géré est documenté dans ce livre — des années de guérilla, un burnout, un fork, une communauté coupée en deux ; à côté, une page de protocole et un médiateur occasionnel sont gratuits. La seconde est empirique : les communs qui durent ont *tous* leurs arènes de conflit, si petites soient-elles — ce n'est pas un luxe de grande organisation, c'est une condition de survie observée sur des siècles ; et l'histoire récente du logiciel libre dit la même chose par la négative, chaque grand projet ayant fini par se doter dans la crise des institutions qu'il refusait dans le calme. La règle du far-west, c'est qu'on y meurt jeune. La question n'est donc pas de savoir si un projet aura des conflits — il en aura, et il doit en avoir, car un projet sans dispute est un projet dont plus personne ne se soucie. La question est de savoir si ses conflits le nourriront ou le dévoreront, et cela ne dépend presque pas des personnes : cela dépend des règles qu'on aura écrites avant. Mais avant d'écrire ces règles, il faut lever un dernier obstacle, et c'est le plus profond : un milieu ne se donne pas d'arènes de conflit tant qu'il croit ne pas en avoir besoin — tant que son image de lui-même lui tient lieu de bilan. Il faut donc examiner cette image, et l'écart entre ce que le logiciel libre dit de lui-même et ce que ses propres archives enregistrent. C'est l'objet du chapitre suivant.

1. La « controverse constructive » : Tjosvold, D. (1985). « Implications of Controversy Research for Management ». *Journal of Management*, 11(3), 21-37 https://doi.org/10.1177/014920638501100303 ; dans la tradition de Deutsch, M. (1973). *The Resolution of Conflict : Constructive and Destructive Processes*. New Haven : Yale University Press https://doi.org/10.12987/9780300159356 — le père de la distinction entre conflit coopératif et conflit compétitif. ↩

2. Jehn, K. A. (1995). « A Multimethod Examination of the Benefits and Detriments of Intragroup Conflict ». *Administrative Science Quarterly*, 40(2), 256-282 https://doi.org/10.2307/2393638 — conflit de tâche contre conflit relationnel. La méta-analyse de De Dreu, C. K. W. & Weingart, L. R. (2003), *Journal of Applied Psychology*, 88(4), 741-749 https://doi.org/10.1037/0021-9010.88.4.741 , montre que même le conflit de tâche nuit à la performance quand il n'est pas contenu — ce qui renforce, plutôt qu'il n'affaiblit, l'exigence de règles et de sécurité psychologique portée ici (Edmondson, chapitre « Le comité n'inventera rien »). ↩

3. Je développe ce partage dans « Image processing does not kill people… and it's a shame » (2021) : les champs techniques capables de nuire — médecine, génie civil — se dotent d'un appareil de diligence (codes déontologiques, normes, permis d'exercer) dont les champs sans risque vital se dispensent, ce qui y « fournit une excuse à la négligence, au manque de rigueur et à la paresse » ; or « les mauvais

outils causent des dommages cognitifs qu'il faudra ensuite défaire », au prix du temps de tous. https://eng.aurelienpierre.com/2021/02/image-processing-does-not-kill-people-and-its-a-shame/ ↩

4. Darktable-org/Darktable, issue #11747 https://github.com/Darktable-org/Darktable/issues/11747, 7 mai 2022 — post-scriptum de la réponse de l'auteur de la refonte (AlicVB) : « I'm sorry to be so rude, but I daily face "that was better before" remarks and injunction in my real work… and that's just nonsense… done by people that (on purpose or not) ignore the reality and how the real world has changed » (notre traduction dans le corps du texte ; archivé : https://web.archive.org/web/20260723092220/https://github.com/Darktable-org/Darktable/issues/11747). Le même message invoque le travail investi et le gel imminent pour refuser tout retour en arrière — angle analysé au chapitre « Le cercle fermé ». Le rapport de bug initial est le mien. ↩

5. La complexité cyclomatique compte les chemins d'exécution linéairement indépendants d'une fonction — chaque embranchement, condition ou boucle, en ajoute un ; c'est en pratique le nombre minimal de cas de test pour couvrir la logique, et une mesure de ce qu'un relecteur doit tenir en tête d'un coup (McCabe, T. J. (1976). « A Complexity Measure ». *IEEE Transactions on Software Engineering*, SE-2(4), 308-320 https://doi.org/10.1109/tse.1976.233837). McCabe proposait de plafonner à 10 par fonction — « limite raisonnable, mais pas magique », notre traduction — et le

guide du NIST n'admet jusqu'à 15 que pour des équipes aguerries, avec relectures organisées et plan de test complet (Watson, A. H. et McCabe, T. J. (1996). *Structured Testing : A Testing Methodology Using the Cyclomatic Complexity Metric.* NIST Special Publication 500-235 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication500-235.pdf, § 2.5, p. 15). Le seuil de 15 est aussi celui de l'outil de mesure employé ici (lizard) : le « plafond » du corps du texte est donc la borne permissive, pas la stricte. Mesures du 23 juillet 2026 sur le dépôt public : à l'étiquette release-3.8.0, src/libs/tools/filter.c compte 548 lignes ; à release-4.0.0, son remplaçant totalise 5 207 lignes (filtering.c : 2 072 ; 16 fichiers sous filters/ : 3 135), l'ancien fichier étant réduit à 127 lignes de délégation. Moyennes par fonction : 1,9 sur 22 fonctions, aucune au-dessus de 15, contre 4,7 sur 147 fonctions, deux au-dessus. Le module collect.c, que la refonte ne remplaçait pas, passait dans le même intervalle de 3 317 à 3 588 lignes. ↩

6. Salon Matrix Darktable-fr, canal de discussion des développeurs francophones, non archivé publiquement : le témoignage relève du régime du chapitre « Écouter sans obéir » et je le cite de mémoire. Ce qui ne relève pas du souvenir : le registre public des fusions du chapitre « Penser contre soi » — d'une quinzaine de fusionneurs de pull requests au début des années 2010 à 2, un seul signant 95 à 100 % des fusions depuis 2019 — et l'arithmétique de relecture du chapitre « Le culte du cargo ». ↩

7. Le rituel se documente parce qu'il est formulaire — relevé par l'API publique du forum principal, juillet 2026. Les mêmes formules reviennent pendant 8 ans, quel que soit le grief. « Nobody is forcing you » : 5 occurrences du même modérateur, de 2018 à 2026. « Fork it » : un développeur de RawTherapee, 2021 — « we don't care much about public opinion… they can fork it and change it, or avoid it » (fil « Exposure tone curves profiles », message 9 https://discuss.pixls.us/t/exposure-tone-curves-profiles-per-brand-or-camera/23367/9) ; un habitué de Darktable, même année, condense le triptyque prouve-le / forke / paie quelqu'un en une seule réponse (fil « Why do CAT and filmic render clipped areas so weirdly », message 81 https://discuss.pixls.us/t/solved-why-do-cat-and-filmic-render-clipped-areas-so-weirdly/26426/81). Deux fils montrent le régime complet. « v4.4 appears grayed-out » (2024) https://discuss.pixls.us/t/v4-4-appears-grayed-out/40694 : un utilisateur signale que le thème par défaut est illisible ; un développeur du cœur répond que ce gris est « a design decision for the best of our user's experience » — irréfutable par construction — et l'invite à ne plus poster (« or maybe you are just a troll ! ») ; Darktable 5.0 livrera un thème à plus fort contraste. « DorS Control Naming Refinement Proposal » (2026) https://discuss.pixls.us/t/dors-control-naming-refinement-proposal/56400 : un contributeur propose de renommer les curseurs d'un module, fait le travail, ouvre une PR. Des habitués lui opposent l'autorité de l'auteur absent : « The author of the module (AP) knew what he was doing ». Cette autorité

s'adosse à mon propre article sur les mathématiques du module — texte falsifiable, lui, mais qui ne dit rien du nom des curseurs. La PR sera retirée. Un gris déclaré irréfutable ; une critique validée par les faits, expulsée quand même ; l'écrit d'un banni brandi comme un dogme sur une question qu'il ne tranche pas : la liturgie est complète. Le même forum sait faire autrement : un fil de 2023 comparant Darktable à Lightroom a reçu une quarantaine de réponses de pure technique et pas une seule éconduite. Le rituel est dominant, pas exclusif. ↩

8. Ostrom, E. (1990). *Governing the Commons*. Cambridge : Cambridge University Press https://doi.org/10.1017/cbo9780511807763 — 6e principe de conception des communs durables : des mécanismes de résolution des conflits rapides, peu coûteux et accessibles localement. ↩

9. Przeworski, A. (2011). « Divided We Stand ? Democracy as a Method of Processing Conflicts ». *Scandinavian Political Studies*, 34(2), 168-182 https://doi.org/10.1111/j.1467-9477.2011.00265.x — conférence du prix Johan Skytte, 2010. La thèse, en 3 temps : la paix civile a presque toujours reposé sur la menace de la force ; or les conflits peuvent être « structurés, régulés et contenus » ; et « des règles purement procédurales peuvent être efficaces pour traiter les conflits » (notre traduction). Du même auteur, la définition la plus courte du régime : « la démocratie est un système où les partis perdent des élections » — *Democracy and the Market* (1991), Cambridge : Cambridge University Press

https://doi.org/10.1017/cbo9781139172493. Les protocoles sont fixes, les issues ne le sont pas. ↩

29. Penser contre soi

Le chapitre précédent a posé que les conflits ont besoin de règles ; celui-ci demande pourquoi un milieu entier vit sans elles depuis 40 ans, et la réponse tient dans son miroir. Le logiciel libre se raconte : ouvert, accueillant, méritocratique, auto-correcteur, sans chefs. Chacun de ces mots est vérifiable, car ce milieu a une propriété unique au monde — tout y est archivé. Chaque forum, chaque dépôt, chaque décision de modération laisse une trace publique et datée. Un mouvement qui prétend vivre de la transparence offre ainsi à l'examen ce qu'aucune institution n'offre : la possibilité de comparer, pièce à pièce, ce qu'il dit de lui-même et ce que ses propres journaux enregistrent. Le test ne s'arrête pas aux manières. Les organisations professent des valeurs qu'elles ne pratiquent pas toujours : la théorie des organisations distingue depuis un demi-siècle la *théorie professée* — ce qu'un groupe déclare croire et vouloir — de la *théorie d'usage* que ses actes révèlent, et l'écart entre les deux est la règle plus que l'exception, souvent sans que l'acteur en ait conscience[1]. Le libre offre à cet écart un banc de mesure unique. « Fait par des photographes, pour des photographes », proclame la page d'accueil du projet — une affirmation sur *qui* il sert, aussi vérifiable que les autres, et que la composition mesurée du milieu dément (chapitre « Ce que l'entre-soi produit ») : fait par des développeurs, pour des développeurs. Le diagnostic commande le remède : un milieu qui ne pense jamais contre lui-même ne peut

pas se corriger, et le chapitre « Le pacte » proposera l'institution qui lui manque.

Le premier article de foi, et le plus célèbre : « avec assez d'yeux, tous les bugs sont superficiels ». La formule n'est pas de Torvalds — c'est Raymond qui l'a forgée et baptisée « loi de Linus », et Torvalds lui-même l'a aussitôt affaiblie, notant que celui qui trouve un problème n'est presque jamais celui qui le comprend[2]. La version affaiblie est raisonnable ; la version slogan a été falsifiée par l'expérience la plus propre qu'on puisse imaginer. La bibliothèque de chiffrement OpenSSL sécurisait une part énorme du web : des millions de serveurs, tous les yeux du monde disponibles. Un défaut de lecture mémoire y fut introduit fin décembre 2011, livré en mars 2012, et découvert en avril 2014 — plus de deux ans d'exposition dans le code le plus stratégique de l'Internet[3]. Et la leçon n'est pas où on la met d'habitude : l'infrastructure n'était pas surveillée par une foule, elle était portée par *un seul* développeur à plein temps, dans une fondation qui recevait environ 2 000 dollars de dons par an. Le trésorier du projet l'a écrit la semaine de la crise : « le mystère n'est pas qu'une poignée de bénévoles surmenés ait raté ce bug ; le mystère est que cela n'arrive pas plus souvent »[3]. Les yeux du dogme n'existent pas : personne ne relit du code de chiffrement gratuitement le samedi. Nadia Eghbal a généralisé le constat en inventoriant la production réelle du logiciel libre : « open source » décrit comment le code est *distribué et consommé* — cela ne dit rien de la façon dont il est *produit* ; la production, mesures à l'appui, repose presque partout sur une ou deux personnes devant un stade de spectateurs[4]. Le bazar est une légende de

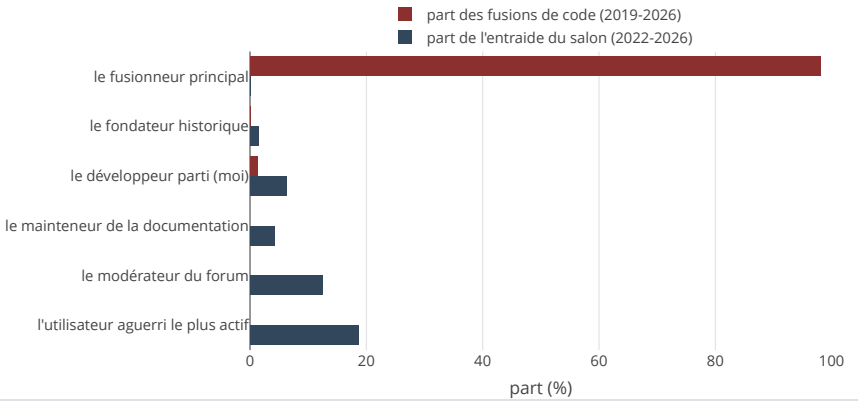
consommateurs. La cathédrale n'a jamais cessé d'être bâtie par 3 ouvriers — non payés.

Deuxième article : la méritocratie. Son premier usage imprimé connu, en 1956, est péjoratif ; le livre qui l'a lancé, deux ans plus tard, est une *dystopie* — une satire où la société du mérite finit renversée par la révolte, et dont l'auteur, Michael Young, a passé sa dernière année à supplier qu'on cesse de brandir positivement le mot qu'il avait inventé pour faire peur[5]. Le logiciel libre l'a adopté comme blason, sans lire le livre : culte du cargo, chapitre « Le culte du cargo », appliqué cette fois à un mot. Reste à tester le contenu du blason. Sa version locale tient en un dogme : seul le code compte, le patch parle de lui-même, peu importe qui le signe. Or ce dogme est mesurable, et il est faux. À contribution comparable, les correctifs des développeurs réputés sont examinés plus vite et aboutissent plus souvent que ceux des inconnus ; et les évaluateurs consultent le profil et les liens sociaux de l'auteur avant de juger son code[6]. Ce qu'on appelle mérite est donc déjà, pour partie, du statut. L'ethnographie a ensuite montré ce que le dogme fait au reste. Dawn Nafus a passé les communautés du libre au crible ; son constat : l'« ouverture » y est appliquée au code et refusée au social. Cette partition délégitime le travail par lequel un nouveau venu, quel qu'il soit, acquiert la compétence : accueil, transmission, parrainage. Le terrain de Nafus était l'exclusion des femmes — d'où son titre, « les patchs n'ont pas de genre », le dogme cité tel quel — mais le mécanisme qu'elle décrit ne choisit pas ses victimes : il écarte quiconque n'est pas déjà dans la place[7]. Le chapitre « Ce que l'entre-soi produit » a donné les chiffres de ce que cela produit ;

on comprend ici le mécanisme qui les protège : un mot-écran, jamais testé, qui transforme toute mesure d'exclusion en constat de mérite.

Troisième article, et le plus profond : « pas de chefs, pas de pouvoir ». Le chapitre « Le comité n'inventera rien » a donné le verdict de Freeman — l'absence de structure ne supprime pas le pouvoir, elle le rend invisible. Il faut maintenant donner la généalogie et la preuve. La généalogie : Robert Michels, 1911, étudiant les partis qui se réclamaient de la démocratie la plus pure, en a tiré ce qu'il a nommé lui-même la « loi d'airain de l'oligarchie » — qui dit organisation, dit *tendance* à l'oligarchie : les élus s'installent, l'appareil se referme, et le pouvoir échappe aux mandants au nom desquels tout se fait[8]. La preuve, un siècle plus tard, sur le terrain même qui prétendait y échapper : Shaw et Hill ont analysé l'historique complet de 683 grands wikis — la production par les pairs dans sa forme chimiquement pure — et testé la loi d'airain en 3 hypothèses : plus une communauté grandit, moins elle nomme de nouveaux administrateurs ; plus les administrateurs concentrent l'activité de gouvernance ; plus les contributeurs expérimentés se font réverter par eux. Les 3 sont vérifiées[9]. Et je n'ai pas besoin d'emprunter le terrain d'autrui : le dépôt de Darktable enregistre sa propre loi d'airain. Le chapitre « Le culte du cargo » en a mesuré le resserrement ; en voici la série complète : 24 personnes fusionnaient du code en 2016, 8 en 2019, deux ou trois depuis 2021 — et un seul développeur signe, chaque année depuis 2019, entre 89 et 100 % de toutes les fusions. La totalité de ce qui entre dans le logiciel passe par les mains d'un seul homme — sans mandat, sans terme, sans appel, et

personne n'a jamais décidé qu'il en serait ainsi : l'organisation l'a sécrété, ce qui est très exactement la thèse de Michels[10]. La conclusion à en tirer n'est pas cynique, elle est architecturale : puisque l'oligarchie pousse *d'elle-même*, le choix n'a jamais été entre pouvoir et absence de pouvoir — il est entre pouvoir responsable et pouvoir irresponsable. Le libre a choisi sans le savoir : des droits de commit sans mandat, des modérateurs sans appel, des vetos sans terme. Le far-west du chapitre « Le cercle fermé » n'était pas une anomalie ; c'était la loi d'airain, opérant dans le noir.



Qui décide, qui prend soin : part des fusions de code du dépôt (2019-2026, 9 316 fusions de pull requests, champ merged_by, rouge) contre part de l'entraide sur le salon de discussion du projet (2022-2026, 9 025 réponses adressées, bleu), par rôle. Deux bases différentes, une question unique — celle du chapitre « Prendre soin » : les mains qui tiennent la décision tiennent-elles le soin ? Le fusionneur principal signe 98 % des fusions et 4 réponses d'entraide en 4 ans et demi ; l'entraide vit chez ceux qui ne décident de rien. Le même fusionneur répondait sur le forum jusqu'en 2019, parmi les 6 premiers, avant de s'en retirer ; et fusionner serait aussi un soin — s'il s'accompagnait de relecture, ce que le chapitre « Le culte du cargo » a mesuré.

Vient alors la question du miroir, et elle se mesure. J'ai fait relever, sur les archives publiques du forum et du projet que ce livre étudie, les principes affichés et les actes enregistrés — en ne retenant que les écarts institutionnels, jamais les personnes, et en consignant aussi les contre-exemples, car il y en a[11]. Trois pièces suffisent ici. La règle affichée du forum ordonne de

« critiquer les idées, pas les gens » et de ne jamais répondre au ton plutôt qu'au fond ; le dossier du chapitre « Le cercle fermé » a montré le régime réel — le fond jamais discuté, le ton discuté sans fin. Le site du projet invite, aujourd'hui encore, dans les mêmes termes qu'en 2022 : « essayez-le, puis dites-nous ce que vous en pensez — rapports de bugs, demandes, suggestions, ou simplement votre opinion » ; les archives montrent ce qu'ont reçu ceux qui ont fait exactement cela — l'accusation de trollage pour un défaut de thème corrigé deux versions plus tard, le « forke et débrouille-toi » pour les autres. Et le jour où un lien fut masqué par la modération, l'explication officielle fut « ce n'était pas censuré » — dans le fil même dont le journal technique enregistre l'acte de masquage, horodaté, signé, puis sa levée le lendemain[11]. Ce dernier cas est le plus instructif, précisément parce qu'il est petit : le lien fut rétabli en un jour, un modérateur s'en est même félicité publiquement — le système *sait* revenir en arrière. Ce qu'il ne sait pas faire, c'est se voir : aucun de ces écarts n'a jamais été constaté *de l'intérieur*. Contre-exemples inclus, les valeurs affichées étaient assez réelles pour être invoquées, et elles ont été appliquées — asymétriquement : avec constance contre les critiques et les étrangers, rarement contre le groupe, jamais contre ceux qui étaient chargés de les faire respecter.

D'où vient l'asymétrie ? Pas de la malice : d'un mécanisme de groupe que la psychologie a nommé et décrit il y a un demi-siècle. Irving Janis, étudiant les fiascos décisionnels des états-majors américains, a isolé la *pensée de groupe* : quand la cohésion devient la valeur suprême, le groupe se met à produire de l'unanimité au lieu de produire du jugement. Ses symptômes se

lisent comme un index du dossier de ce livre : l'illusion d'unanimité — le chapitre « Le cercle fermé » a montré la spirale de silence qui la fabrique ; l'autocensure — les dissidents d'IRC qui désapprouvaient en privé et se taisaient en public ; la pression directe sur qui doute ; et les « gardiens de la pensée », le mot est de Janis — *mindguards*, sur le modèle du garde du corps : des membres auto-désignés qui protègent le groupe non des coups, mais des informations qui ébranleraient sa confiance en lui-même[12]. Relisez la modération du chapitre « Le cercle fermé » avec ce mot-là. Et mesurez alors l'ironie, car elle est la clef du chapitre : ce mouvement est né pour combattre des dominations réelles — l'éditeur propriétaire qui dicte à l'utilisateur ce qu'il a le droit de faire, le logiciel espion qui prélève ses données. Il a gagné ces combats dans le code. Et il a recréé, dans ses forums, la structure exacte qu'il combattait : un pouvoir qui dicte les croyances admissibles, surveille la conformité et éteint la dissidence — la dictature chassée du code, réinstallée dans la conversation. Ce que le groupe demande à ses membres n'est ni un besoin à servir ni un soin à donner — le chapitre « Ce que l'entre-soi produit » a montré que ces mots-là manquent à l'éthos ; ce qu'il demande, c'est la conformité : aimer le flux de travail, louer l'outil, adopter le jargon, et acclamer la nouveauté au rythme de l'économie d'acclamation.

Qu'ont les institutions qui ne tombent pas dans ce piège ? Elles ne sont pas peuplées de gens meilleurs ; elles sont *équipées contre elles-mêmes*. La science est l'exemple canonique, et il faut le lire correctement. Merton a décrit son éthos en 4 normes, dont la dernière, le *scepticisme organisé*, dit tout dans son adjectif : le

doute n'y est pas une vertu individuelle, c'est un mandat institutionnel — « le chercheur ne préserve pas le clivage entre le sacré et le profane, entre ce qui exige le respect non critique et ce qui peut être objectivement analysé »[13]. Rien, pas même les croyances fondatrices de l'institution, n'est exempté d'examen. Popper a tiré la conséquence qui renverse le sens commun : l'objectivité scientifique ne résulte pas — ne *peut pas* résulter — des efforts d'un chercheur pour être objectif ; elle résulte de la « coopération amicalement hostile » de beaucoup de chercheurs, c'est-à-dire d'institutions : revues par les pairs, réplication, droit de réfutation[14]. Une société ouverte, au sens de Popper, n'est pas une société sans croyances : c'est une société dont les croyances restent attaquables de l'intérieur. Le logiciel libre a ouvert son code et fermé sa culture — tout de l'artefact est inspectable, rien de l'institution ne l'est : pas d'audit de modération, pas d'instance d'appel, pas de bilan public des conflits, pas une seule enquête que le milieu se soit infligée à lui-même. Son unique organe de correction est le *fork* — c'est-à-dire la sortie. Or Hirschman a démontré le vice de ce choix : la sortie facile n'améliore pas le système, elle l'anesthésie, car ceux qui tiennent le plus à la qualité — « les agents les plus actifs, les plus fiables et les plus créatifs de la prise de parole » — sont précisément ceux qui partent les premiers, et leur départ prive la critique interne de ses porteurs[15]. « Forke » n'est pas une politique de correction : c'est l'externalisation de la conscience. Et le dossier du chapitre « Le cercle fermé » montre le geste qui complète le circuit : le forkeur, conscience externalisée, est ensuite excommunié — le milieu se débarrasse de sa critique en le priant de partir, tout en conservant ses contributions, puis se félicite de ne plus l'entendre.

Reste à dire ce que serait l'ouverture véritable, et c'est la thèse de ce chapitre. Le mouvement a ouvert le code — c'est-à-dire qu'il a ouvert le pouvoir à ceux qui savent le lire. Le chapitre « La liberté dans le vide » a chiffré ce que cette condition exclut : l'immense majorité des utilisateurs, pour qui « la liberté de modifier le programme » est une clause écrite dans une langue qu'ils ne parleront jamais. On ne peut pas exiger des utilisateurs qu'ils lisent le code ; on peut leur ouvrir autre chose : la *gouvernance*. Des sièges, des mandats, des comptes rendus, un droit d'appel — une structure sociale bâtie par-dessus le code ouvert, qui est restée le principal angle mort de l'entre-soi précisément parce que l'entre-soi n'a jamais eu à y penser : entre pairs qui lisent le code, l'ouverture technique suffit. Ce n'est pas une utopie sans précédent ; c'est une tradition que le libre n'a jamais regardée. Au début des années 1980 — avant le manifeste GNU —, le projet UTOPIA réunissait à Stockholm des chercheurs et le syndicat nordique des ouvriers du livre : des typographes, 120 000 membres derrière eux, co-concevaient dans un « laboratoire de technologie » leurs propres outils de mise en page, non parce qu'ils savaient programmer, mais parce que le travail était le leur — la conception participative, codifiée depuis en discipline académique, a démontré il y a 40 ans que des non-programmeurs peuvent gouverner la conception de leurs outils[16]. L'ironie veut que ce fussent des ouvriers de l'imprimerie : le métier même dont le libre a hérité ses polices, ses casses et son vocabulaire. Voilà donc la prochaine ouverture, et elle n'est pas technique : élargir aux utilisateurs non pas le texte du programme, mais le gouvernement de sa conception — et pour le milieu, apprendre enfin à penser contre soi, c'est-à-dire cesser d'être un club pour

devenir une institution. La forme concrète de cette institution — qui siège, qui paie, qui tranche, qui répond — occupe les deux chapitres qui ferment cette partie : le « travailleur incompétent », où la maison apprend à reconnaître et à borner le savoir-faire qui lui manque sans chasser personne ; puis le « pacte » qui la fonde.

---

1. Argyris, C. & Schön, D. A. (1974). *Theory in Practice : Increasing Professional Effectiveness*. San Francisco : Jossey-Bass — la distinction entre *espoused theory* (théorie professée : ce qu'un acteur déclare croire et poursuivre) et *theory-in-use* (théorie d'usage : ce que ses actes révèlent effectivement), et le constat que l'écart entre les deux est ordinaire et souvent inaperçu de l'acteur lui-même. ↩

2. Raymond, E. S. (1999). *The Cathedral and the Bazaar*. Sebastopol, CA : O'Reilly (exposé initial : Linux Kongress, Wurtzbourg, 27 mai 1997 ; trad. fr. de S. Blondeel, « La cathédrale et le bazar », en ligne) http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/ . Leçon 8 : « Étant donné un vivier de bêta-testeurs et de co-développeurs suffisamment grand, presque chaque problème sera vite caractérisé et sa solution évidente pour quelqu'un » ; « ou, moins formellement : "avec assez d'yeux, tous les bugs sont superficiels". Je baptise cela : loi de Linus ». Raymond note lui-même l'objection de Torvalds : celui qui trouve le problème n'est pas, en général, celui qui le comprend. ↩

3. CVE-2014-0160 (« Heartbleed ») https://nvd.nist.gov/vuln/detail/CVE-2014-0160 : défaut introduit le 31 décembre 2011, livré dans OpenSSL 1.0.1 le 14 mars 2012, divulgué le 7 avril 2014. Le témoignage financier : Marquess, S. (2014). « Of Money, Responsibility, and Pride ». veridicalsystems.com, 12 avril 2014 (archivé) http://web.archive.org/web/20140415221341/http://veridicalsystems.com:80/blog/of-money-responsibility-and-pride/ — environ 2 000 $ de dons par an ; un seul membre de l'équipe sans emploi extérieur à plein temps (Stephen Henson) ; « il devrait y avoir au moins une demi-douzaine de membres à plein temps, pas un seul » ; « le mystère n'est pas qu'une poignée de bénévoles surmenés ait raté ce bug ; le mystère est que cela n'arrive pas plus souvent » (notre traduction). La fondation vivait par ailleurs de contrats commerciaux (moins d'un million de dollars bruts par an), liés à des livrables, non à la maintenance du cœur. ↩ ↩

4. Eghbal, N. (2020). *Working in Public : The Making and Maintenance of Open Source Software*. San Francisco : Stripe Press https://press.stripe.com/working-in-public : « le terme "open source" ne décrit que la façon dont le code est distribué et consommé. Il ne dit rien de la façon dont il est produit » (notre traduction). Elle y rapporte qu'une étude sur GitHub attribue, dans plus de 85 % des projets examinés, plus de 95 % du code et des interactions à moins de 5 % des développeurs ; et sa typologie du « stade » : beaucoup d'usagers, presque pas de contributeurs. En amont : Eghbal, N. (2016). *Roads and Bridges : The Unseen Labor Behind Our*

Digital Infrastructure. Ford Foundation https://www.fordfoundation.org/learning/library/research-reports/roads-and-bridges-the-unseen-labor-behind-our-digital-infrastructure/ . ↩

5. Premier usage imprimé recensé (et première citation de l'*Oxford English Dictionary*) : Fox, A. (1956). « Class and Equality ». *Socialist Commentary*, mai 1956 — déjà péjoratif. Le livre : Young, M. (1958). *The Rise of the Meritocracy, 1870-2033 : An Essay on Education and Equality*. Londres : Thames & Hudson https://archive.org/details/riseofmeritocracooooyoun (trad. fr. *La Méritocratie : mai 2033*, trad. M. Luciani, Paris : SÉDÉIS, coll. « Futuribles », 1969) — satire dystopique s'achevant sur la révolte contre l'élite du mérite. Le désaveu : Young, M. (2001). « Down with meritocracy ». *The Guardian*, 29 juin 2001 https://www.theguardian.com/politics/2001/jun/29/comment — « j'ai été tristement déçu par mon livre de 1958 » ; Young y déplore l'adoption du mot, jusque dans les discours de Tony Blair, comme idéal positif. ↩

6. Bosu, A. et Carver, J. C. (2014). « Impact of Developer Reputation on Code Review Outcomes in OSS Projects : An Empirical Investigation ». *Proceedings of ESEM 2014*, ACM/IEEE https://doi.org/10.1145/2652524.2652544 — sur 8 projets libres majeurs : les correctifs des développeurs du cœur reçoivent un premier retour plus vite et achèvent leur revue plus tôt que ceux des contributeurs périphériques. Sur l'usage des signaux sociaux dans l'évaluation : Marlow, J.,

Dabbish, L. et Herbsleb, J. (2013). « Impression Formation in Online Peer Production ». *Proceedings of CSCW 2013*, 117-128 https://doi.org/10.1145/2441776.2441792 ; et Tsay, J., Dabbish, L. et Herbsleb, J. (2014). « Influence of Social and Technical Factors for Evaluating Contribution in GitHub ». *Proceedings of ICSE 2014* https://doi.org/10.1145/2568225.2568315 — l'acceptation d'une contribution croît avec la proximité sociale de son auteur au projet, à facteurs techniques donnés. ↩

7. Nafus, D. (2012). « "Patches don't have gender" : What is not open in open source software ». *New Media & Society*, 14(4), 669-683 https://doi.org/10.1177/1461444811422887 — l'« ouverture » y est analysée comme une construction qui sépare le technique du social : les hommes « monopolisent la signature du code et délégitiment du même geste les liens sociaux nécessaires pour construire des mécanismes d'inclusion » (notre traduction). ↩

8. Michels, R. (1911). *Zur Soziologie des Parteiwesens in der modernen Demokratie*. Leipzig : Klinkhardt https://archive.org/details/zursoziologiedeoomichgoog (trad. fr. : *Les Partis politiques : essai sur les tendances oligarchiques des démocraties*, trad. S. Jankélévitch, Paris : Flammarion, 1914 — « qui dit organisation, dit tendance à l'oligarchie », p. 15-16 ; traduction intégrale moderne : *Sociologie du parti dans la démocratie moderne*, trad. J.-C. Angaut, Paris : Gallimard, coll. « Folio essais », 2015). La formule non modalisée (« who says organization, says oligarchy », éd. anglaise 1915, p. 401) est plus dure que l'original allemand,

qui parle de *tendance* ; c'est la version modalisée qui est citée ici. ↩

9. Shaw, A. & Hill, B. M. (2014). « Laboratories of Oligarchy ? How the Iron Law Extends to Peer Production ». *Journal of Communication*, 64(2), 215-238 https://doi.org/10.1111/jcom.12082 — l'historique interne complet de 683 des plus grands wikis de la plateforme Wikia, 2004-2010 ; les 3 hypothèses (raréfaction des nominations d'administrateurs, concentration de l'activité de gouvernance, réversions croissantes des contributeurs expérimentés) sont toutes trois vérifiées : « nous trouvons un soutien à la loi d'airain de Michels ». Limites énoncées par les auteurs : une seule plateforme, l'oligarchie opérationnalisée par le comportement administratif. ↩

10. Propositions de code fusionnées du dépôt Darktable-org/Darktable, relevé exhaustif par l'API (champ `merged_by`), 2012-2026 : 11, 15, 16, 13, 11, 8, 6, 5, 6, 4, 3, 3, 2, 3 puis 2 fusionneurs distincts par an ; part des deux premiers : 34 à 65 % de 2012 à 2015, 70 % en 2016, 99 % dès 2019, 100 % à l'unité près depuis 2023. Une mesure antérieure sur les seuls commits de fusion donnait le même profil avec des effectifs plus élevés avant 2019 (jusqu'à 24) : les fusions locales de branches, sans pull request, y comptaient comme des fusions — la mesure par pull request est la bonne pour la question posée, qui est le droit de trancher les contributions d'autrui. Sur un échantillon de 21 propositions fusionnées en 2026, les 21 le furent par le même

développeur. Le chapitre « Le culte du cargo » fait l'arithmétique de ce que ce goulot rend impossible. ↩

11. Registre constitué sur archives publiques (API Discourse du forum, journaux d'action inclus ; https://www.Darktable.org/development/ , formulation vérifiée identique entre l'instantané Wayback du 3 juillet 2022 et la page vivante de 2026, réarchivée : https://web.archive.org/web/20260613134148/https://www.Darktable.org/development/). Les 3 pièces : règles du forum, https://discuss.pixls.us/faq (« criticize ideas, not people ») ; l'invitation du projet et son accueil réel, dont https://discuss.pixls.us/t/40694 (accusation de trollage pour un signalement de thème, corrigé ensuite ; archivé : https://web.archive.org/web/20260714060151/https://discuss.pixls.us/t/v4-4-appears-grayed-out/40694) ; le masquage journalisé, https://discuss.pixls.us/t/interview-with-ap/31924 (archivé : https://web.archive.org/web/20250513031948/https://discuss.pixls.us/t/interview-with-ap/31924) — `action_code : visible.disabled` (31 juillet 2022), « it was not censored » (1er août 2022), levée du masquage le même jour. Les contre-exemples sont consignés au même registre : rétablissement rapide et salué, une intervention de modération en faveur du critique, des reprises de code créditées ; et le forum n'a jamais prétendu être une plateforme de libre expression — ses responsables l'ont dit sans détour, ce qui borne l'accusation à ce qui est documenté. ↩ ↩

12. Janis, I. L. (1972). *Victims of Groupthink*. Boston : Houghton Mifflin https://openlibrary.org/books/OL21279371M/Victims_of_groupthink — le « mindguard », p. 41-42 : « de même qu'un garde du corps protège des agressions physiques, un gardien de la pensée protège [le groupe] des pensées qui pourraient ébranler sa confiance dans le bien-fondé des politiques auxquelles il est engagé » (notre traduction) ; la typologie des 8 symptômes en 3 familles : Janis, I. L. (1982). *Groupthink*, 2e éd. Boston : Houghton Mifflin, p. 174-175 https://openlibrary.org/books/OL3513757M/Groupthink. Pas de traduction française publiée ; « pensée de groupe » est l'usage établi. ↩

13. Merton, R. K. (1942). « Science and Technology in a Democratic Order ». *Journal of Legal and Political Sociology*, 1, 115-126 ; repris comme « The Normative Structure of Science », in *The Sociology of Science*. Chicago : University of Chicago Press, 1973, chap. 13, p. 267-278 https://archive.org/details/sociologyofscienoooomert_j7p1 — les 4 normes (universalisme, « communisme » des résultats, désintéressement, scepticisme organisé), p. 270 ; le sacré et le profane, p. 277-278. ↩

14. Popper, K. R. (1945). *The Open Society and Its Enemies*, vol. II, chap. 23 (« The Sociology of Knowledge »). Londres : Routledge https://www.routledge.com/The-Open-Society-and-its-Enemies-Hegel-and-Marx/Popper/p/book/9780415278423 — « l'objectivité scientifique ne résulte pas (et ne peut pas résulter) des

efforts d'un chercheur individuel pour être objectif, mais de la coopération amicalement hostile de nombreux chercheurs » (notre traduction ; p. 217-218 de l'édition Routledge révisée). En français : *La Société ouverte et ses ennemis*. Paris : Seuil, 1979 (version abrégée) ; traduction intégrale : Paris : Les Belles Lettres, 2025. ↩

15. Hirschman, A. O. (1970). *Exit, Voice, and Loyalty*. Cambridge, MA : Harvard University Press https://www.hup.harvard.edu/books/9780674276604 (déjà convoqué au chapitre « Le cercle fermé »), p. 43 : « la présence de l'option de sortie peut tendre à atrophier le développement de l'art de la prise de parole » ; p. 47 : ceux qui tiennent le plus à la qualité, « agents les plus actifs, les plus fiables et les plus créatifs de la prise de parole », sont ceux qui sortent les premiers ; p. 51 : leur départ « paralyse la prise de parole en la privant de ses agents principaux » (notre traduction). ↩

16. Bødker, S., Ehn, P., Kammersgaard, J., Kyng, M. & Sundblad, Y. (1987). « A UTOPIAN Experience : On Design of Powerful Computer-Based Tools for Skilled Graphic Workers », in Bjerknes, G., Ehn, P. & Kyng, M. (dir.), *Computers and Democracy — A Scandinavian Challenge*. Aldershot : Avebury, p. 251-278 https://search.worldcat.org/title/15489500. Rétrospective : Sundblad, Y. (2011). « UTOPIA : Participatory Design from Scandinavia to the World ». *History of Nordic Computing 3*, IFIP AICT 350. Berlin : Springer, p. 176-186 https://doi.org/10.1007/978-3-642-23315-9_20. Le projet

(1981-1986, Stockholm et Aarhus) associait le syndicat graphique nordique (~120 000 membres) à la conception d'outils de mise en page et de traitement d'image. La somme doctrinale : Ehn, P. (1988). *Work-Oriented Design of Computer Artifacts*. Stockholm : Arbetslivscentrum https://books.google.com/books/about/Work_oriented_Design_of_Computer_Artifac.html?id=dxCJswEACAAJ ; le recueil de référence : Schuler, D. & Namioka, A. (dir.) (1993). *Participatory Design : Principles and Practices*. Hillsdale, NJ : Erlbaum https://www.routledge.com/Participatory-Design-Principles-and-Practices/Schuler-Namioka/p/book/9780805809510 . ↩

30. Le travailleur incompétent

Le chapitre « Le travailleur invisible » a montré un bout de la chaîne : le contributeur compétent qui porte l'ouvrage sans qu'on le voie, jusqu'à l'épuisement. Voici l'autre bout, le plus embarrassant à nommer : le contributeur dont le manque de savoir-faire ne peut être ni dit, ni corrigé, ni écarté. L'incompétence dont il s'agit n'est pas d'abord celle d'une personne : c'est celle d'une institution. Le milieu n'a aucun moyen de traiter la compétence — ni de la filtrer à l'entrée, ni de l'évaluer en chemin, ni de remédier à son manque. Le travailleur incompétent n'est pas un sot qu'on moque ; c'est le symptôme d'une maison qui ne sait ni qui elle laisse entrer, ni ce que vaut son travail, ni comment dire à quelqu'un de partir. Décrire ce vide, c'est dessiner en creux l'institution qui manque à tout projet recruté sur le seul enthousiasme.

D'où vient qu'un projet fourmille de gens qui ne savent pas faire le travail ? De ce qu'il est deux choses à la fois : une école et un produit. On y vient pour apprendre — pour le plaisir, pour le défi, pour un savoir-faire à porter ensuite sur un CV. La recherche a nommé ces motifs et les a mesurés : le contributeur signale son talent à de futurs employeurs et pairs[1] ; il apprend le métier en participant d'abord depuis la marge, selon le modèle de l'apprentissage situé[2] ; et les enquêtes de motivation placent l'acquisition de compétences parmi les premières raisons de contribuer[3]. Rien de tout cela n'est illégitime — c'est même

précieux. Mais une école et un produit ont des fonctions de qualité opposées. Dans une école, l'erreur est la méthode : on apprend en se trompant. Dans un produit dont un public dépend, l'erreur est le dommage : on la traque et on la hait. Le logiciel libre confond les deux, et fait tourner le produit comme une école. Bon pour l'apprenti, mauvais pour le produit : voilà la confusion, énoncée.

Une organisation ordinaire filtre à l'entrée — elle embauche —, évalue en cours de route — elle passe le travail en revue — et peut séparer — elle licencie. Le libre n'a aucun des trois : on ne choisit pas ses collègues, personne n'a évalué leurs compétences, et rien ne permet de s'en séparer. Ce n'est pas un oubli, c'est un schéma récurrent. Elinor Ostrom, en cherchant ce qui fait durer les communs, a dégagé des principes de conception ; trois d'entre eux sont exactement ceux qui manquent ici. Des frontières claires : qui est membre, qui a des droits. Une surveillance : la conduite et la contribution de chacun sont observées. Des sanctions graduées : des conséquences réelles, jusqu'à l'exclusion[4]. Le libre n'a ni frontière — la bienveillance fait de tout arrivant un membre —, ni surveillance — évaluer une compétence passe pour une agression. Quant à la sanction, elle existe, mais retournée. Faute de pouvoir renvoyer quelqu'un franchement, on l'ignore, ou on l'use : le silence sur ses contributions, la question sans fin, la réponse qui ne vient jamais. Le milieu a remplacé le licenciement — honnête, borné, notifié — par l'ostracisme et l'épuisement, qui ne sont rien de tout cela. C'est pire, et c'est plus hypocrite : on ne dit pas à quelqu'un qu'il n'est pas à la hauteur, on le laisse le comprendre en se cognant au vide. Là où le

commun durable a une institution d'appartenance — franche sur qui entre, sur ce que vaut le travail, sur qui doit partir —, le projet n'a qu'un tourniquet ouvert et une porte dérobée par où l'on pousse les gens dehors sans le dire.

Car la compétence, ici, ne peut pas se dire — alors qu'elle se définit sans peine. Concevoir, le chapitre « Le mot et la chose » l'a posé avec Simon, c'est transformer une situation existante en situation préférée : réduire l'écart entre ce qui est et ce qu'on voudrait. La compétence est alors la propriété d'un individu qui rétrécit cet écart, ne serait-ce qu'un peu, sans abîmer ce qui l'entoure. Les deux clauses comptent. La première rejoint la plus ancienne définition psychologique de la compétence — la capacité d'interagir efficacement avec son milieu[5] — et ce que McClelland opposait dès 1973 au culte du diplôme : ce qui atteste la compétence, c'est la performance réelle, jamais le titre qui la préjuge[6]. La seconde est celle que le libre oublie : réduire un écart en en creusant d'autres n'est pas une compétence, c'est un déplacement de dette. Celui qui corrige un bug en cassant trois modules, qui livre une fonction en alourdissant la maintenance de tous, qui gagne un utilisateur en en rebutant dix, n'a pas rapproché la situation préférée : il a déplacé l'écart ailleurs. Et cet environnement à ne pas abîmer par l'acte de conception ne se limite pas au produit à livrer, mais s'étend au collectif, à la structure sociale, au reste du vivant et aux écosystèmes. C'est le nécessaire garde-fou contre le résultat quoi qu'il en coûte. La compétence se juge sur un solde, pas sur un geste. Ce solde est un fait vérifiable : la qualité d'un correctif, le même défaut réintroduit trois fois, la leçon jamais tirée en cinq ans. Mais

l'énoncer est reçu comme une offense de caractère — de l'arrogance, de la brutalité. C'est le partage du chapitre « Le conflit bien tempéré » reporté sur les personnes : un jugement de compétence est un jugement de vérité, et un milieu sans métrique du vrai ne sait l'entendre que comme une affaire de ton. Il n'existe alors aucun protocole par lequel « cette décision a été prise par quelqu'un qui n'était pas en mesure de la juger » puisse être dit et admis. L'incompétence devient indicible ; indicible, elle devient ingérable. On ne corrige pas ce qu'on n'a pas le droit de constater.

Et l'interdit est d'autant plus absurde que l'incompétence est le plus facile à établir — bien plus que la compétence. Le bien fait est difficile à prouver : il est ouvert, cumulatif, souvent invisible — le bug évité ne laisse aucune trace, comme le chapitre « Compter les absents » l'a montré. Le mal fait, lui, se voit : une régression, un module cassé, un utilisateur perdu suffisent à le signaler. La psychologie en a fait une loi — *le mauvais est plus fort que le bon* : plus saillant, plus constant, plus vite perçu que son équivalent positif[7]. L'ingénierie de la fiabilité en vit : on ne mesure pas la « bonté » d'un système, on compte ses défaillances, et la sûreté s'est longtemps définie par l'absence d'accident avant qu'on n'essaie, bien plus difficilement, de mesurer ce qui va bien — le partage Safety-I / Safety-II du chapitre « La racine et le symptôme ». C'est enfin la structure même de la falsifiabilité, fil de tout ce livre : on ne prouve pas une compétence, il y faudrait une infinité d'actes réussis ; on la réfute d'un seul acte nuisible. La compétence est un pôle qu'on n'atteint jamais ; l'incompétence, un fait qu'un cas établit. La caractériser par le mal fait n'est donc

pas seulement plus commode : c'est la seule prise falsifiable — et c'est pourquoi ce chapitre porte le nom qu'il porte.

Une réserve aussitôt, car mesurer par le seul mal a son revers : qui ne juge que sur les dégâts récompense celui qui ne fait rien — le plus sûr moyen de ne rien casser est de ne rien tenter. C'est pourquoi la mesure reste un *solde* : le mal se lit sur la colonne du débit, mais il se pèse contre le bien de la colonne du crédit. On n'écarte pas le maladroit qui prend des risques utiles et apprend de ses erreurs ; on écarte le nuisible net — celui dont le débit dépasse durablement le crédit, sans redressement en vue.

Reste à dire comment se fait la somme, puisqu'une faute et une contribution ne sont pas de même monnaie. Il n'existe pas de taux de change naturel entre un module cassé et une conception heureuse ; réduire les deux à un même chiffre — un « score de compétence » — serait la fausse précision que le chapitre « Mesurer sans se mentir » a bannie. La sociologie l'a montré : cette réduction, la *commensuration*, n'est jamais neutre — c'est un acte social qui décide de ce qui compte et efface le reste[8]. Le solde ne se calcule donc pas d'un trait ; il se lit en deux temps. D'abord des planchers : certaines nuisances disqualifient quel que soit l'apport — celles que la définition a déjà mises hors jeu, l'atteinte aux utilisateurs, au collectif, au vivant, le résultat obtenu « quoi qu'il en coûte ». Aucune brillance ne les rachète ; ce ne sont pas des poids dans une balance, mais des valeurs qu'on refuse d'y mettre — ce que la psychologie de la décision nomme des valeurs *protégées*[9]. Au-dessus du plancher seulement, apports et torts ordinaires se pèsent l'un contre l'autre ; et là, le poids relatif de chacun est un choix de valeurs, non un fait

technique. Il doit donc être *déclaré*, avec le reste des valeurs du chapitre « Le pacte », et non improvisé au cas par cas selon qui l'on juge. Le juge mandaté l'applique ; la falsifiabilité rend son application vérifiable. On ne fabriquera pas la formule parfaite — une part du poids reste affaire de jugement, et le jugement ne s'automatise pas. Mais un barème déclaré qu'on peut contester vaut infiniment mieux qu'une balance secrète tenue par la main la plus puissante.

Le prix de l'incompétence se paie deux fois. D'abord, former est un travail — réel, non payé, et déversé sur les seuls qui savent déjà, c'est-à-dire sur le travailleur invisible déjà submergé : chaque arrivant qu'il faut mettre à niveau est une taxe prélevée sur celui qui peut le moins la supporter. Et cette taxe ne se rembourse pas : l'apprenti formé s'en va souvent une fois acquis ce qu'il était venu chercher. Le savoir-faire tient sur un CV, il se transporte ; la maintenance de ce qu'il laisse derrière lui, non — elle reste, orpheline. La communauté forme à ses frais des gens qui capitalisent ailleurs le fruit de cette formation, et hérite du code qu'ils n'entretiendront plus. Ensuite, un savoir-faire qui n'est sélectionné nulle part crée des problèmes qui retombent sur les sujets compétents, qui, lassés, finissent par s'en aller — la maison se vide (chapitre « Ce que l'entre-soi produit »), et ce qui reste se resserre en médiocratie (chapitre « Le cercle fermé »). Il faut pourtant retourner le tableau, car l'arrivant n'est pas que du coût. Il voit ce que le vétéran ne voit plus : que la documentation est illisible, que l'accueil est hostile, que l'interface est opaque pour qui découvre. Le « superflu » que l'ancien dédaigne — un guide, un tutoriel, une entrée plus douce — est souvent sa

contribution la plus réelle, et le réflexe méritocratique qui lui refuse voix au chapitre rend le projet aveugle à ses propres angles morts. La bonne réponse n'est donc pas de fermer la porte : un milieu qui n'admet que des experts déjà formés se prive de son seul regard neuf, et se condamne à rester illisible pour tout autre que lui. Former quelqu'un est d'ailleurs du soin au même titre que servir un utilisateur : du travail donné à qui ne peut pas encore, par qui peut déjà. Et comme tout soin, il demande à être organisé et payé plutôt que prélevé en douce sur celui qui sait — faute de quoi la maison forme jusqu'à épuisement de ses formateurs.

Le deuxième prix de l'incompétence se paie si le travail de formation a échoué, ou si la formation n'a pas été demandée, c'est à dire si le défaut de compétence n'a pas été reconnu et nommé, aussi bien par son sujet que par son superviseur : c'est le report du travail mal fait sur les collègues. Celui qui fait mal, déplace le problème ou l'aggrave au lieu de le résoudre, voire crée des problèmes par erreur de jugement ou par négligence, fait porter à tous le prix de ses erreurs. Et dans mon expérience du monde libre, cette incompétence est tabou : on ne peut jamais dire de quelqu'un qu'il est incompétent, peu importe le niveau de preuve qui pèse contre les conséquences de ses erreurs répétées, c'est directement une insulte. L'absence de rémunération couplée à la bonne volontée semble absoudre absolument toutes les fautes et toutes les leçons non apprises, peu importe le nombre de gens qui vont en pâtir.

Reste la question qui fâche : mesurer la compétence, soit, mais comment, et par qui ? Le libre a déjà ses métriques de prestige, et

elles sont toutes superficielles. Le volume — le nombre de commits, l'ancienneté, la présence sur les canaux — récompense l'activité, pas sa valeur : on peut produire beaucoup et n'améliorer rien. Le ton — la civilité, la conformité à l'entre-soi — récompense l'agrément, pas la justesse : on peut être charmant et se tromper toujours. Ni l'un ni l'autre ne dit si le travail est bon. La seule mesure qui le dise est falsifiable : la capacité, vérifiable, d'améliorer le projet. Et « améliorer » n'est ni vague ni seulement technique — c'est mieux couvrir les besoins des utilisateurs, concevoir et documenter l'outil pour qu'il serve des gens réels, bâtir une infrastructure qui tient, rendre plus saines les méthodes et la structure sociale du projet. La compétence se mesure à ce qu'une contribution fait avancer, sur l'un quelconque de ces plans, au regard de critères posés d'avance. Un tel critère a une vertu que le volume et le ton n'ont pas : il est vérifiable par un tiers — n'importe qui peut reprendre le dossier et refaire le compte. Mais vérifier n'est pas décider, et c'est là que les vraies questions commencent.

Sur quoi, d'abord, lit-on la compétence ? Sur les résultats — mais les mesures objectives d'un travail et les jugements qu'on porte sur lui ne se recouvrent qu'à moitié : aucun ne suffit seul[10], et tout indicateur chiffré, sitôt qu'il devient la cible, se corrompt, comme l'a établi le chapitre « Mesurer sans se mentir ». Sur la trajectoire — la vitesse à laquelle quelqu'un apprend et cesse de répéter ses erreurs, ce que la recherche sur la pratique délibérée sait mesurer (chapitre « L'instrument et la machine »), mais qui exige un historique et une base de comparaison. Sur le jugement des pairs — qui saisit ce que le chiffre manque, mais que le

chapitre « Penser contre soi » a montré biaisé par la réputation et les signaux sociaux. Aucune de ces prises n'est sûre seule ; il faut les croiser. Et même croisées, elles ne renferment pas tout : une part de la compétence est tacite, sue sans pouvoir se dire[11], et aucun barème ne l'épuise.

Reste la question que j'avais esquivée : un critère falsifiable ne s'applique pas tout seul. Quelqu'un le relève, l'interprète et tranche — accorde un droit, ouvre une formation, prononce une exclusion. Ce quelqu'un est un chef, et le nier rejouerait l'illusion que le chapitre « La liberté dans le vide » a défaite : l'absence de structure ne supprime pas le pouvoir, elle le cache. Il y a déjà un chef, de fait — le fusionneur dont le veto ne se conteste pas. La falsifiabilité ne le fait pas disparaître ; elle rend ses décisions contestables, et c'est toute la différence. Le chef de fait juge la compétence sans critère et sans appel — « tais-toi et corrige » ; un chef mandaté la jugerait sur des critères qu'un tiers vérifie, par une décision qu'on peut renverser en montrant, pièces en main, qu'elle est fausse. Sur quoi nommer ce chef ? Non sur l'ancienneté ni le volume de fusions, mais par le collectif qu'il affecte, pour un mandat borné, et révocable : la démocratie sur qui détient l'autorité, le métier sur la façon de l'exercer — l'assemblée du chapitre « Le pacte ». Le pouvoir de juger la compétence est inévitable ; le rendre falsifiable, mandaté et révocable est tout ce qui sépare un chef d'un tyran. On ne bâtira jamais le compétencemètre parfait ; on peut faire qu'y avoir tort se corrige, et qu'y être injuste se conteste. C'est peu, mais c'est exactement ce qui manque.

Entre l'exclusion par l'arrogance et la tolérance sans limites, il y a ce qu'a toute institution durable : un protocole d'appartenance, et falsifiable. Déclarer, à l'entrée, ce que le projet suppose qu'on sache faire — une attente explicite, adossée aux valeurs affichées du chapitre « Le pacte ». Asseoir le prestige et l'appartenance sur la mesure d'amélioration qui précède, non sur le volume ni sur le ton. Proportionner l'autorité à la compétence démontrée, et la faire croître avec elle : les droits de fusion, le poids dans une décision se gagnent sur pièces et s'étendent à mesure que le dossier s'étoffe — l'arrivant décide dans un périmètre borné, l'expert dans un périmètre large. Ouvrir un chemin de formation à qui veut apprendre, dont le coût est assumé et partagé, non abandonné à un seul. Réserver l'exclusion — bornée, notifiée, susceptible d'appel, les sanctions graduées d'Ostrom — à qui ne peut ni ne veut, et la prononcer franchement plutôt que par le vide. Elle n'est pas la punition d'un individu : elle est la garantie que l'impact d'un seul ne devienne pas la peine de tous. Un contributeur qu'on ne borne pas ne fait pas que mal travailler dans son coin ; il déverse ses erreurs sur ceux qui devront les réparer et sur les utilisateurs qui les subiront. Écarter, alors, c'est protéger le collectif d'une externalité, non châtier une personne — et c'est pourquoi cela peut, et doit, se faire sans mépris. Et surtout, séparer l'aire d'apprentissage de la production : que le débutant apprenne sans que ses erreurs atteignent le public qui dépend de l'outil. Que l'école soit une école, dans un espace qui n'est pas le produit. Rien là n'exige de la cruauté ; tout exige une structure.

Le travailleur incompétent n'a jamais été la cible. La cible est une structure incapable et de faire grandir le savoir-faire, et d'en borner l'absence — et qui, faute des deux, finit par chasser ceux qui savent. Le travailleur invisible rencontré plus tôt et le travailleur incompétent de ce chapitre-ci sont les deux victimes d'une même institution manquante : le premier s'épuise de ce que le second ne peut ni apprendre ni quitter. Nommer le second sans le premier serait une cruauté ; les nommer ensemble, c'est nommer le vide qui les produit.

---

1. Lerner, J. & Tirole, J. (2002). « Some Simple Economics of Open Source ». *Journal of Industrial Economics*, 50(2), 197-234 https://doi.org/10.1111/1467-6451.00174 — l'*incitation au signal* (*career concern*) : contribuer à du code public et visible signale sa compétence à de futurs employeurs et à ses pairs, motif économique central de la participation au libre. ↩

2. Lave, J. & Wenger, E. (1991). *Situated Learning : Legitimate Peripheral Participation*. Cambridge : Cambridge University Press https://doi.org/10.1017/cbo9780511815355 — on apprend un métier en participant d'abord à la périphérie d'une communauté de pratique ; le modèle décrit l'entrée type dans un projet libre, et rappelle qu'une telle communauté est d'abord un lieu d'apprentissage. ↩

3. Sur l'apprentissage et l'acquisition de compétences (*human capital*) comme motifs dominants : Lakhani, K. R. & Wolf, R. G. (2005). « Why Hackers Do What They Do », in Feller, J. *et al.* (dir.), *Perspectives on Free and Open Source Software*.

Cambridge, MA : MIT Press https://doi.org/10.7551/mitpress/5326.003.0005 ; Hars, A. & Ou, S. (2002). « Working for Free ? Motivations for Participating in Open-Source Projects ». *International Journal of Electronic Commerce*, 6(3), 25-39 https://doi.org/10.1080/10864415.2002.11044241 . ↩

4. Ostrom, E. (1990). *Governing the Commons*. Cambridge : Cambridge University Press https://doi.org/10.1017/cbo9780511807763 — parmi les huit principes de conception des communs durables : (1) des frontières clairement définies (l'appartenance : qui est membre) ; (4) une surveillance des comportements ; (5) des sanctions graduées, jusqu'à l'exclusion. Le chapitre « Le conflit bien tempéré » en a déjà invoqué le sixième (des mécanismes de résolution des conflits accessibles). ↩

5. White, R. W. (1959). « Motivation Reconsidered : The Concept of Competence ». *Psychological Review*, 66(5), 297-333 https://doi.org/10.1037/h0040934 — la compétence y est définie comme la capacité d'interagir efficacement avec son environnement, source d'une motivation propre (l'*effectance*). ↩

6. McClelland, D. C. (1973). « Testing for Competence Rather Than for "Intelligence" ». *American Psychologist*, 28(1), 1-14 https://doi.org/10.1037/h0034092 — contre le culte des tests et des diplômes : ce qui atteste la compétence est la performance démontrée en situation, non le titre ou le score censés la préjuger. ↩

7. Baumeister, R. F., Bratslavsky, E., Finkenauer, C. & Vohs, K. D. (2001). « Bad is Stronger than Good ». *Review of General Psychology*, 5(4), 323-370 https://doi.org/10.1037/1089-2680.5.4.323 ; Rozin, P. & Royzman, E. B. (2001). « Negativity Bias, Negativity Dominance, and Contagion ». *Personality and Social Psychology Review*, 5(4), 296-320 https://doi.org/10.1207/s15327957pspr0504_2 — les événements négatifs ont des effets plus grands, plus constants et plus vite perçus que les positifs de même ampleur : le mal se détecte mieux que le bien. ↩

8. Espeland, W. N. & Stevens, M. L. (1998). « Commensuration as a Social Process ». *Annual Review of Sociology*, 24, 313-343 https://doi.org/10.1146/annurev.soc.24.1.313 — la commensuration, réduction de qualités hétérogènes à une mesure commune, n'est pas une opération neutre : c'est un acte social et politique qui rend certaines choses comparables et en efface d'autres. ↩

9. Baron, J. & Spranca, M. (1997). « Protected Values ». *Organizational Behavior and Human Decision Processes*, 70(1), 1-16 https://doi.org/10.1006/obhd.1997.2690 — des valeurs que les gens refusent de mettre en balance avec d'autres biens, tenues pour non négociables quel que soit le gain offert en échange. ↩

10. Bommer, W. H. *et al.* (1995). « On the Interchangeability of Objective and Subjective Measures of Employee Performance : A Meta-Analysis ». *Personnel Psychology*, 48(3), 587-605 https://doi.org/10.1111/j.1744-

6570.1995.tb01772.x — les mesures objectives de performance (les résultats) et les évaluations subjectives (les jugements portés sur le même travail) ne corrèlent que modérément : elles ne sont pas interchangeables, aucune ne se substitue à l'autre. ↩

11. Polanyi, M. (1966). *The Tacit Dimension*. Garden City, NY : Doubleday — « nous pouvons savoir plus que nous ne pouvons dire » (*we can know more than we can tell*) : une part du savoir-faire est *tacite*, mobilisée sans pouvoir être pleinement explicitée ni réduite à des règles. ↩

31. Le pacte

Ce livre a consacré 5 parties à déconstruire ; il doit à son lecteur désormais de proposer la reconstruction. Tous les défauts des chapitres précédents ont une racine commune, et le chapitre « La liberté dans le vide » l'a nommée : le logiciel libre a pensé la liberté et jamais les conditions de la liberté — ni la capacité de l'usager, ni les conditions du travailleur, ni la structure du collectif. On ne répare pas cela par plus de liberté ; on le répare en fournissant ce que la liberté seule ne fournit pas, et ce chapitre décrit l'instrument qui le ferait — un instrument qui n'est pas dessiné pour un seul projet : il est à prendre, tel quel ou amendé, par n'importe quelle communauté qui veut durer. Le reste du livre s'appuyait sur des mesures ; celui-ci s'appuie sur un raisonnement, cohérent avec ce que la recherche sait des communs.

Le diagnostic économique commande le reste. Le logiciel libre fait coexister deux transactions à sens unique, et c'est leur conjonction qui empoisonne tout. La première : l'usager reçoit le produit sans rien donner en retour — non par malice, mais parce que 3 siècles de marché nous ont appris qu'une chose dématérialisée, téléchargeable à l'infini, n'a pas de valeur intrinsèque, donc pas de prix. La seconde, symétrique : le développeur décide sans rendre de comptes — la licence ayant déclaré qu'il ne devait rien à personne, il ne doit rien, pas même une explication. D'un côté un produit pris sans contrepartie, de

l'autre un pouvoir exercé sans responsabilité : ce ne sont pas deux problèmes, c'en est un seul, vu de ses deux bouts. Le milieu a d'ailleurs déjà son pacte, il suffit de l'écouter s'énoncer : « c'est le contrat implicite du logiciel libre — vous recevez le travail des autres, et vous remboursez avec votre temps », écrivait un développeur du cœur de Darktable dès 2016, avant de conclure que les usagers qui ne contribuent pas « sont un fardeau, pas un actif »[1]. Voilà le contrat existant, énoncé par une partie : l'usager doit ; le développeur, lui, ne doit rien — la licence l'en a dispensé. Un contrat dont une seule partie a des obligations ne s'appelle pas un contrat, mais une domination, et la résoudre par une domination symétrique (le développeur tenu de livrer les modifications exigées par les utilisateurs) ne suffirait pas à l'annuler. Le pacte proposé ici ne fait que rendre les deux flèches explicites, et réciproques.

Il faut maintenant poser la responsabilité morale que tout ceci esquive. Quand un projet de loisir devient assez répandu pour que des gens en dépendent — des utilisateurs qui ont investi des années d'apprentissage, des bibliothèques entières de retouches, leur flux de travail et parfois leur gagne-pain —, il naît une responsabilité de faire que ce projet réponde aux besoins de ceux qui en dépendent. Je dis bien *une* responsabilité, et non *la* responsabilité du développeur initial : rien n'oblige celui qui a ouvert le chantier à le porter seul, et le chapitre « Le travailleur invisible » a montré ce qu'il en coûte de l'exiger de lui. C'est une responsabilité de la *communauté* — celle-là même qui s'autorise aujourd'hui un double discours : d'un côté « le logiciel libre vous libère des géants du numérique », de l'autre « les développeurs ne

vous doivent rien ». Les deux phrases ne peuvent pas être vraies ensemble : on ne recrute pas des dépendants au nom de la liberté pour leur opposer ensuite l'exemption au nom du bénévolat. Assumer cette responsabilité exige des structures — c'est tout ce chapitre — mais d'abord un changement de regard : voir le logiciel libre comme un *service public*, au sens plein — un service rendu à un public qui en dépend, avec la continuité et la responsabilité que le mot exige —, et non comme une ruche d'initiatives individuelles qui survivront, ou non, selon l'humeur et la disponibilité de leurs auteurs. La dépendance des usagers est un fait, déjà acquis ; la seule question encore ouverte est de savoir si elle sera assumée par une institution, ou exploitée par une absence d'institution. Le travailleur invisible du chapitre « Le travailleur invisible » s'épuise dans l'écart entre les deux, et le chapitre « La violence à trois étages » a montré ce que le pouvoir sans contrepartie produit, dedans comme dehors. Tout le reste — l'entre-soi, l'agressivité, le mépris de l'utilisateur — pousse sur ce terrain-là, alimenté par une frustration symétrique des uns envers les autres.

Un service se juge aussi à sa fidélité à sa mission déclarée. Un projet affiche des valeurs : pour qui il conçoit, ce qu'il refuse, ce qu'il promet. Encore faut-il que l'affichage soit honnête, et qu'il le reste. Trois exigences en découlent, qui prolongent la règle de falsifiabilité de tout ce livre. *Déclarer* ces valeurs explicitement, plutôt que les laisser en slogan d'accueil. Les rendre *vérifiables* : l'écart entre la valeur professée et la théorie d'usage — qui l'on sert vraiment, ce que l'on fusionne, ce que l'on refuse — doit pouvoir se mesurer, comme le chapitre « Penser contre soi » l'a

fait contre les archives d'un projet. Les rendre *lisibles à l'entrée*, enfin : un nouveau venu doit pouvoir confronter les valeurs du projet aux siennes avant d'y engager des années. La gestion des ressources humaines l'a mesuré : un aperçu réaliste du poste — donné avant l'engagement, ses servitudes autant que ses attraits — réduit les départs ultérieurs, parce qu'il aligne les attentes sur la réalité[2] ; et la congruence entre les valeurs de la personne et celles de l'organisation prédit la satisfaction, l'engagement et le maintien[3]. Le prospectus honnête n'est donc pas une politesse : c'est un filtre d'entrée qui épargne les défections coûteuses. Sans lui, on ne découvre le désaccord qu'une fois la sortie devenue chère — l'entrée non éclairée est la cause cachée des départs tardifs du chapitre « Le travailleur invisible ». Je l'ai éprouvé : entré sur la foi d'un tel prospectus — « fait par des photographes, pour des photographes » —, j'ai mis des années à mesurer la dérive entre l'affiche et les actes, et le constat n'a rien eu d'agréable. Un projet qui déclare faussement ses valeurs ne trompe pas seulement ses usagers : il recrute ses contributeurs sur un prospectus qu'il n'honore pas.

La sortie n'est pas de choisir un camp mais de refermer les deux flèches en un cercle : faire que l'usager contribue et que le producteur réponde, non par vertu, mais parce que la structure les lie. C'est très exactement ce qu'est une coopérative — un communisme ancien, éprouvé et qui fonctionne, où l'entreprise, donc le projet, est possédée par ceux qui l'utilisent et par ceux qui y travaillent, partageant le pouvoir de vote. Le pacte est là, et il est de responsabilité mutuelle : les producteurs répondent des besoins des usagers ; les usagers répondent des conditions de

travail des producteurs — un environnement sûr, équitable, et de quoi vivre. Ni le développeur-dictateur à qui l'on devrait une gratitude éternelle, ni l'usager-sangsue qui réclame beaucoup et donne peu : deux classes de membres d'une même maison, chacune tenue à l'autre. C'est le contraire exact du far-west du chapitre « Le cercle fermé » : là où il n'y avait aucun tribunal, la coopérative institue une assemblée ; là où le conflit se réglait par les droits d'accès, il se règle par le vote et le mandat.

Le partage des légitimités est la clé de voûte, et il reprend le fil tendu depuis le chapitre « Écouter sans obéir » jusqu'au chapitre « Le comité n'inventera rien » : le collectif est souverain sur les *buts* et les *budgets*, le métier est souverain sur les *moyens*. Concrètement, l'assemblée générale sécurise un budget annuel — le plus souvent par les cotisations des membres —, en décide l'emploi, et hiérarchise les priorités de haut niveau : améliorer le masquage, le travail par lots, la prise en charge de tel format. Elle ne tranche pas l'implémentation, ni le détail de la conception, qui se prêtent mal au vote et relèvent du chapitre « Poser le problème ». Les développeurs y défendront les chantiers de fond — la refonte d'une dette technique, pour réduire le coût de maintenance futur ; les usagers, les fonctionnalités qui leur manquent : l'assemblée délibère et classe, elle n'exécute pas. Ce partage résout d'un coup ce que les chapitres précédents opposaient : il donne à l'usager le pouvoir qui manquait au chapitre « Le cercle fermé » — celui de peser sur la direction, voire de démettre une équipe — sans lui donner celui de dessiner à la place du concepteur, que le chapitre « Écouter sans obéir » lui a

refusé. La démocratie sur les fins, le métier sur les moyens : la ligne n'est jamais franchie, dans aucun des deux sens.

Une objection peut venir, cette fois, du camp du soin lui-même. L'éthique du care se méfie des contrats : Virginia Held a montré que la pensée contractuelle généralise le cas d'égaux indépendants qui n'existent nulle part — le chapitre « La liberté dans le vide » s'en est servi — et elle se défiait tout autant d'une morale qui se juridicise[4]. Un pacte écrit, des mandats, des votes : n'est-ce pas retomber dans le travers ? C'est ici que l'objet et l'échelle décident. Held parlait du droit qui colonise les relations de soin à l'échelle d'une société ; le pacte ci-dessus opère dans un collectif de quelques dizaines de personnes où l'alternative au droit écrit n'a jamais été le soin — ça a été le far-west mesuré au chapitre « Le cercle fermé ». Et sa propre conclusion tranche : la justice, écrivait-elle, doit être logée *dans* le cadre plus large des relations de soin, non l'inverse. C'est très exactement la construction de ce chapitre — non pas remplacer le soin par du droit, mais donner au soin les quelques règles opposables sans lesquelles il reste un slogan. Reste l'avertissement que l'histoire adresse à tout cadre de soin qui réussit. La définition des soins infirmiers de Virginia Henderson et sa grille des besoins fondamentaux, enseignées à chaque génération d'infirmières, ont fini aussi en grilles d'évaluation de la dépendance et en forfaits de facturation — l'instrument d'émancipation devenu instrument comptable[5]. Un pacte qui vieillit bien n'est pas celui qui ne bouge pas : c'est celui dont l'assemblée relit les clauses en se demandant qui elles servent encore. Le jour où ce pacte servira la

structure plutôt que les personnes, il faudra le réviser dans l'esprit où il fut écrit — contre l'usure, pas contre les gens.

La coopérative d'usagers-sociétaires règle le problème que les dons ponctuels n'ont jamais réussi à régler dans les projets libres. C'est mesuré — les études longitudinales sur le financement participatif du logiciel montrent un effet réel à court terme et nul à long terme, et le chapitre « Le travailleur invisible » a dit pourquoi : le donateur est aussi l'usager, ce qui entretient un rapport de consommateur, pas de copropriétaire, et l'on ne finance pas une recherche après coup, comme on ne finance pas la NASA une fois la fusée partie. La cotisation coopérative est d'une autre nature que le don : elle n'achète pas un produit déjà livré, elle acquiert une part de la maison et une voix dans ses choix ; elle se verse avant, parce qu'elle finance un projet dont on est copropriétaire, non un service ou un produit dont on attend qu'il ait fait ses preuves avant de s'engager. C'est le déplacement décisif — du pourboire reconnaissant à la cotisation constitutive.

Le pacte a aussi une portée qui déborde le confort, et l'actualité l'a rendue littérale. En 2024, une porte dérobée s'est glissée dans une bibliothèque présente sur presque tous les systèmes du monde, non par un exploit technique mais par l'ingénierie sociale d'un mainteneur solitaire et épuisé, poussé pendant des années à céder sa place à son futur attaquant. Le chapitre « Le travailleur invisible » en tirait une exigence de sécurité ; le pacte en est la réponse structurelle. Un mainteneur isolé, non payé, sans contre-pouvoir, est une vulnérabilité de la chaîne d'approvisionnement mondiale ; un travailleur tenu par une structure qui le rémunère, l'entoure et partage la décision ne l'est pas au même degré. Le

confort du travailleur, on l'a dit, n'est pas un supplément d'âme ; ici il devient une mesure de sûreté publique, et le pacte, son instrument.

Reste l'objection de fond : la preuve manque — on ne peut pas citer la grande coopérative de logiciel libre qui aurait démontré le modèle. Trois choses tempèrent ce constat sans l'annuler. D'abord, l'instrument juridique existe déjà : la société coopérative d'intérêt collectif, en droit français, est précisément une structure à parties prenantes multiples qui lie des membres bénéficiaires et des membres salariés — le pacte n'est pas à inventer, il est à appliquer. Ensuite, la trajectoire que je propose est celle que la recherche observe : les projets qui réussissent sont gouvernés de façon légère et informelle au petit format — un responsable qui fait, une vision claire —, et ne se dotent d'institutions formelles qu'en grandissant ; ce que ce livre a paru opposer, l'homme seul aujourd'hui et la coopérative demain, est une seule et même route, prise à deux moments — celle qu'Elinor Ostrom a dégagée en étudiant les communs qui durent, où les institutions formelles n'arrivent qu'avec l'échelle[6]. Enfin, l'histoire du logiciel libre est pleine de collectifs qui ont fini par se donner la structure qu'ils niaient — une constitution élue après une crise de succession, un conseil de direction après un fiasco, un code de conduite imposé jusqu'au plus solitaire des dictateurs bienveillants[7]. « Pas de structure » n'a jamais été un état, seulement un délai avant la structure ; le pacte propose de ne pas attendre la crise pour l'écrire.

Le pacte referme cette partie, et il referme plus qu'elle. Le livre a ouvert sur une définition : concevoir, c'est résoudre le problème

d'un humain, dont on a choisi de faire une lecture technique. Le pacte en tire la conséquence la plus lointaine. L'humain dont il s'agit n'est pas seulement l'utilisateur au bout de la chaîne : c'est aussi celui qui fait le travail, et c'est la communauté entière qui les tient ensemble. Le cahier des charges, la mesure, le moteur, la simplicité, l'instrument, la structure : tout ce que ce livre a défendu tient dans une seule idée, que le pacte porte à sa forme collective — derrière chaque problème technique, il y a un humain. Le pacte donne au mot « communauté », si facilement brandi dans le milieu logiciel, son sens premier : un groupe de gens qui prennent soin les uns des autres, et qui s'engagent les uns envers les autres, et non pas juste un groupe d'intérêt qui s'enthousiasme pour les mêmes choses.

Reste à regarder cette idée en face. La discipline entière de ce livre a un nom, que l'ingénierie ne prononce jamais ; la conclusion existe pour le prononcer.

---

1. https://discuss.pixls.us/t/975 (mars 2016, archivé : voir chapitre « Ce que l'entre-soi produit ») : « that's the implicit contract that comes with free software : You get the work of others to use and pay back with your time to report problems and help debugging them. Freeloaders are a burden and not an asset » (notre traduction). ↩

2. Phillips, J. M. (1998). « Effects of Realistic Job Previews on Multiple Organizational Outcomes : A Meta-Analysis ». *Academy of Management Journal*, 41(6), 673-690 https://doi.org/10.2307/256964 — l'*aperçu réaliste du poste*

(présenter au candidat les contraintes autant que les avantages, avant l'embauche) abaisse le renouvellement du personnel et rapproche les attentes de la réalité vécue. ↩

3. Kristof-Brown, A. L., Zimmerman, R. D. & Johnson, E. C. (2005). « Consequences of Individuals' Fit at Work : A Meta-Analysis of Person-Job, Person-Organization, Person-Group, and Person-Supervisor Fit ». *Personnel Psychology*, 58(2), 281-342 https://doi.org/10.1111/j.1744-6570.2005.00672.x — l'adéquation personne-organisation, dont la congruence de valeurs est le cœur, prédit la satisfaction au travail, l'engagement et le maintien dans le poste. ↩

4. Held, V. (2006). *The Ethics of Care : Personal, Political, and Global*. New York : Oxford University Press — sur la généralisation contractuelle, p. 81 ; sur la place de la justice : « caring relations should form the wider moral framework into which justice should be fitted », p. 71 (notre traduction dans le corps du texte) https://doi.org/10.1093/0195180992.001.0001 . ↩

5. Le modèle de Virginia Henderson (grille des 14 besoins fondamentaux) structure le recueil de données infirmier français ; le matériel de cours officiel des IFSI enseigne que ce modèle « a influencé […] la prise en charge financière des soins de santé », grille AGGIR à l'appui — l'évaluation de la perte d'autonomie qui conditionne l'allocation dépendance (support de cours, IFSI de Troyes https://www.ifsi-troyes.fr/sites/default/files/le_modele_conceptuel_decrit_par_virginia_henderson_2.pdf) ; le Bilan de Soins Infirmiers,

assis sur la même logique de besoins, tarife aujourd'hui les soins infirmiers à domicile en forfaits journaliers. ↩

6. Sur l'émergence de la légitimité : O'Mahony, S. & Ferraro, F. (2007). « The Emergence of Governance in an Open Source Community ». *Academy of Management Journal*, 50(5), 1079-1106 https://doi.org/10.5465/amj.2007.27169153 (étude longitudinale de Debian). Sur la trajectoire légère-puis-formelle : Schweik, C. M. & English, R. C. (2012). *Internet Success*. Cambridge, MA : MIT Press https://doi.org/10.7551/mitpress/9780262017251.001.0001 . Cadre des communs, tel que formulé dans Ostrom, E. (1990). *Governing the Commons*. Cambridge : Cambridge University Press https://doi.org/10.1017/CBO9780511807763 (trad. fr. *Gouvernance des biens communs*, Bruxelles : De Boeck, 2010) — dont le 6e principe est l'existence de mécanismes de résolution des conflits accessibles et peu coûteux. Cadre des communs adapté au logiciel (où le risque n'est pas la surexploitation d'Ostrom mais la sous-fourniture et l'épuisement) : Frischmann, B. M., Madison, M. J. & Strandburg, K. J. (dir.) (2014). *Governing Knowledge Commons*. Oxford : Oxford University Press https://doi.org/10.1093/acprof:oso/9780199972036.001.0001 ; en amont, Benkler, Y. (2002). « Coase's Penguin, or, Linux and *The Nature of the Firm* ». *Yale Law Journal*, 112(3) https://doi.org/10.2307/1562247 . Le cadre juridique français : société coopérative d'intérêt collectif, loi n° 2001-624 du 17 juillet 2001 https://www.legifrance.gouv.fr/jorf/id/JORFTEXT000000757800 . ↩

7. Exemples publics, cités par le rôle des institutions et non des personnes : l'adoption par un grand langage de programmation d'un conseil de direction élu après la démission de son dictateur bienveillant historique (2018) ; la constitution votée d'un grand système d'exploitation communautaire (1998), avec chef élu et résolutions générales comme garde-fou ; l'institution d'un conseil de gouvernance dans un jeune langage système après une crise (2023), dont le post-mortem impute la crise à l'absence de « règles et processus clairs » ; l'adoption d'un code de conduite par le plus centralisé des noyaux (2018). « Méritocratie » proclamée et « pushyocratie » réelle : Nafus, D. (2012). « "Patches don't have gender" ». *New Media & Society*, 14(4) https://doi.org/10.1177/1461444811422887. ↩

Conclusion — Prendre soin

La partie technicienne de ce livre peut se résumer à ceci : la conception est une discipline qu'on s'impose partout où notre inclination naturelle — la paresse — tend à nous faire privilégier à court terme nos propres intérêts de concepteur : la solution rapide, à peine définie, qui n'écoute pas les besoins de l'utilisateur (soit qu'elle les ignore par condescendance, soit qu'elle se fasse aveugler par « ce que veut l'utilisateur » littéralement), qui s'arrête trop tôt dans la remontée des causes du problème, qui regarde sous le lampadaire. La technique n'en est qu'une grille de lecture choisie, pas la finalité.

La partie sociale peut se résumer au fait qu'aucun objet technique n'est jamais neutre et n'existe hors du groupe qui le crée, et que la question n'est pas de savoir s'il appartient à ses concepteurs ou à ses utilisateurs : il appartient à tous, et il doit pouvoir répondre aux besoins des uns sans épuiser les autres en maintenance. Ainsi la conception de la structure sociale qui produit l'artefact technique est-elle une partie de la conception de l'artefact lui-même, aussi bien au niveau de la gouvernance qu'au niveau de l'organisation du travail. Ici, il faut autant refuser les discours qui légitiment l'oppression de l'utilisateur que ceux qui tentent de construire une vision christique du concepteur, ignorant ses propres besoins, sa sécurité et son confort.

La partie propre au logiciel libre enterre définitivement les fausses croyances des libristes sur eux-mêmes : ils n'émancipent que des privilégiés, et leurs structures sociales prétendument horizontales sont des dictatures qui se retirent la possibilité de penser leurs mécanismes de domination, tout en glissant vers des comportements sectaires et dogmatiques qui réduisent au silence aussi bien les critiques de l'intérieur que celle de l'extérieur.

La partie « La liberté et ses conditions » s'est fermée sur un pacte ; cette conclusion prononce en face ce que le prologue a annoncé et que le pacte, comme tout le livre avant lui, cherchait geste après geste. Écouter avant de décider, caractériser le besoin avant la solution, tester la réception au lieu de la présumer, maintenir au lieu de larguer, répondre de ce qu'on livre : chapitre après chapitre, j'ai décrit une discipline. Son vrai nom n'appartient pas au vocabulaire de l'ingénierie, et il est temps de le regarder en face : prendre soin.

Le mot détonne. La conception telle que nous la pratiquons vient de l'ingénierie, l'ingénierie vient de la machine, et la machine vient d'un monde d'hommes. Ce n'est pas une figure de style : l'histoire des techniques a documenté la construction délibérée de l'ingénieur comme figure masculine — au tournant du vingtième siècle, le métier s'est défini en excluant, et l'exclusion fait partie de la définition[1]. Un siècle plus tard, l'empreinte est intacte. Aux États-Unis, les femmes obtiennent un quart des diplômes d'ingénieur ; le génie mécanique, discipline la plus grosse en volume et matrice historique des méthodes de conception industrielle, en est à 18 % — et l'électricité comme l'informatique font pire[2]. Le logiciel libre, qui se raconte comme l'utopie

ouverte de ce monde-là, en est la caricature : 1,5 % de femmes parmi les développeurs au début des années 2000, 3 % dans la grande enquête de 2017, et selon la méthode de mesure — enquête déclarative ou inférence sur les registres de commits — entre 8 et 14 % aujourd'hui. Jamais, en 20 ans, l'ordre de grandeur n'a dépassé le septième[3]. Le milieu qui écrit les outils de tout le monde est, socialement, le plus étroit de tous les métiers techniques. Ce n'est pas une remarque de morale : c'est une donnée de conception. Un milieu homogène conçoit pour lui-même, la partie « La liberté et ses conditions » a passé 4 chapitres à mesurer ce que cela produit, et le présent chapitre dit ce qui manque en face.

Ce qui manque porte un nom dans la philosophie morale, et il se trouve que ce nom a été forgé par celles que l'ingénierie avait exclues. L'éthique du *care* — le mot anglais s'est imposé, le français dit le soin — naît au début des années 1980 chez Carol Gilligan, qui montre qu'à côté de la morale des principes et des droits il existe une morale de la relation et de la responsabilité concrète, entendue d'abord comme une « voix différente »[4] ; chez Nel Noddings, qui décrit l'attention absorbée par l'autre et le déplacement de motivation vers ses besoins — ce que ce livre a appelé écouter le témoignage plutôt que la prescription[5] ; puis chez Joan Tronto, qui arrache le soin au registre des vertus féminines pour en faire une catégorie politique. Virginia Held en a tiré la théorie morale complète, et son théorème est celui que ce livre retrouve par la mesure : il peut exister du soin sans justice ; il n'existe pas de justice sans soin, car sans soin personne ne survit pour être respecté[6]. La définition de Tronto et Berenice

Fisher décrit exactement notre métier : le soin est « une activité générique qui comprend tout ce que nous faisons pour maintenir, perpétuer et réparer notre “monde” », de sorte que nous puissions y vivre aussi bien que possible[7]. Maintenir, perpétuer, réparer un monde pour qu'on puisse y vivre : qu'est-ce que la conception (en particulier logicielle), sinon cela ? Tronto décompose le soin en phases — se soucier, prendre en charge, donner le soin, recevoir le soin, et plus tard prendre soin *ensemble* — et la correspondance avec ce livre est terme à terme : se soucier, c'est poser le problème (chapitre « Poser le problème ») ; prendre en charge, c'est spécifier et s'engager (chapitres « Mesurer sans se mentir » et « Livrer, pas larguer ») ; donner le soin, c'est construire ; recevoir le soin, c'est tester la réception réelle au lieu de la présumer (chapitre « Compter les absents ») — la phase que l'ingénierie oublie toujours, car elle suppose que le destinataire du soin a voix au chapitre ; et le soin *ensemble*, c'est la gouvernance que les chapitres « Ouvrir sans tenir le pinceau » et « Le pacte » ont dessinée[8]. La conception est une activité de soin. Elle a été inventée, historiquement, par le seul milieu qui refusait de la penser comme telle — et la recherche a fini par faire la jonction : l'éthique du soin a été portée tardivement dans les sciences et techniques, et jusque dans l'interaction homme-machine, où un agenda féministe du design existe noir sur blanc depuis 2010[9]. Même la littérature du design n'admet pourtant le mot « soin » que là où il touche l'hôpital : le seul livre grand public qui ose « Design as Caregiving » est un livre sur les systèmes de santé[10]. Le soin serait pour les malades ; les bien-portants, eux, n'auraient droit qu'à des fonctionnalités.

Or c'est précisément l'inverse, et voici le cœur de ma thèse. Le monde se technicise, et il se technicise sans demander l'avis de ceux qui le subissent. Ellul l'a dit il y a 70 ans : la technique s'auto-augmente, s'impose comme milieu, et ne passe jamais par le vote[11]. Langdon Winner l'a dit d'une image : nous traversons en somnambules la reconstruction des conditions mêmes de l'existence humaine[12]. Les techniques sont poussées du haut vers le bas, par ceux qui savent et qui peuvent, sur ceux qui ont besoin et qui endurent. Personne n'a voté pour le guichet dématérialisé, pour l'application obligatoire, pour la fin du formulaire papier ; on a décrété que c'était le progrès, et « on n'arrête pas le progrès » n'est pas un constat : c'est un permis. Un permis de contourner la discussion démocratique sur les techniques que nous voulons, celles que nous acceptons, et les conséquences que nous tolérons qu'elles aient sur nos vies — l'historien David Noble a montré ce que cette rhétorique de l'inévitable recouvre de choix très évitables, faits par très peu de gens[13]. Et quand la philosophie de la technique cherche l'antidote, elle le trouve du côté des usagers qui interviennent dans la conception même : ce qu'Andrew Feenberg appelle la rationalisation démocratique — les profanes forçant la technique à intégrer leurs intérêts contre la hiérarchie technicienne[14]. C'est, en langage de philosophe, exactement le programme du chapitre « Ouvrir sans tenir le pinceau » et du pacte.

Dans un monde ainsi artificialisé de force, l'illettrisme numérique n'est pas une faiblesse individuelle : c'est une blessure infligée par le milieu. La recherche sur la fracture numérique le répète depuis 20 ans : l'accès ne suffit pas, les compétences sont socialement

distribuées, et le non-usage est un fait de structure, pas un défaut de caractère[15]. Norman a documenté le mécanisme intime : face à un outil mal conçu, l'utilisateur ne blâme pas l'outil, il se blâme lui-même, jusqu'à l'impuissance apprise — il cesse d'essayer[16]. Voilà l'état du destinataire. Et que lui répond le logiciel libre, qui se présente comme sa libération ? Le mépris. L'illettré numérique y est traité en fardeau ; le nouveau venu y affronte des barrières sociales documentées — réponses hostiles, condescendance, jargon-mot-de-passe — que la recherche a cataloguées jusque dans le vocabulaire des listes de diffusion[17] ; la culture du « lis donc le manuel » y tient lieu d'accueil, et la partie « La liberté et ses conditions » a montré le club que cela construit. Le chapitre « La violence à trois étages » a donné le verdict de ce mépris : quand l'« erreur humaine » est presque toujours une erreur de conception, mépriser l'utilisateur en échec, c'est mépriser son propre travail mal fait. Celui qui a le plus besoin de soin est traité comme celui qui en mérite le moins.

C'est ici que l'analogie médicale cesse d'être une image. La médecine aide les êtres vivants à vivre : elle réduit la souffrance, allège les maux et les entraves, restaure le mouvement et le confort — soulager et guérir, « obligations jumelles » de la profession, dit un texte classique[18]. La conception aide les êtres *technicisés* à vivre dans un monde artificialisé : elle réduit la friction, allège les procédures, restaure la capacité d'agir de celui que la technique avait réduit à l'impuissance. Bernard Stiegler a donné à cette parenté son concept : la technique est un *pharmakon*, à la fois « ce qui permet de prendre soin et ce dont il faut prendre soin — puissance curative dans la mesure et la

démesure où c'est une puissance destructrice »[19]. Le concepteur est le pharmacien de ce poison-remède : le même objet technique qui émancipe, mal dosé, mal pensé, détruit — et il n'y a personne d'autre que nous entre le dosage et le patient.

Si la conception est la médecine du monde artificiel, alors il lui faut ce que la médecine a : un serment, et une déontologie. L'idée n'est pas neuve. Hippocrate d'abord, dans le texte réel plutôt que dans la légende — la maxime n'est pas dans le Serment, elle est dans les Épidémies : « avoir, dans les maladies, deux choses en vue : être utile, ou du moins ne pas nuire »[20]. Victor Papanek ensuite, qui ouvrait en 1971 son livre sur la responsabilité sociale du designer par la phrase la plus insolente de la discipline — il existe des métiers plus nuisibles que le design industriel, mais très peu[21]. Mike Monteiro, plus près de nous, réclame pour les designers un code d'éthique et une responsabilité professionnelle organisée sur le modèle de la médecine — la formule « serment d'Hippocrate du designer » est celle de ses commentateurs, mais l'argument est bien le sien[22]. Le logiciel libre lui-même a vu naître une « licence hippocratique », interdisant l'usage d'un logiciel pour violer les droits humains — et l'establishment de l'open source l'a rejetée au nom de la définition canonique, qui protège la « liberté zéro » de nuire[23] : jusque dans ses licences, le milieu a choisi l'abstention de soin. Voici donc, pour finir, le serment que je formule, et qui condense le livre. *Être utile, ou du moins ne pas nuire* : ne pas détruire le capital de celui qui m'a fait confiance — ses données, ses années d'apprentissage, son flux de travail (chapitre « Livrer, pas larguer »). Ne pas faire porter le coût de mon confort de développeur par ceux qui n'ont rien demandé

(chapitre « Travailler pour le travail »). Concevoir pour celui qui ne pourra pas se défendre contre ma conception : l'illettré, le pressé, le fatigué — l'utilisateur en état de faiblesse est le cas nominal, pas le cas limite (chapitre « La violence à trois étages »). Écouter le témoignage avant de décider, et tester la réception après avoir décidé, parce que le soin qui ne vérifie pas s'il a été reçu n'est pas un soin, c'est une émission (chapitres « Poser le problème » et « Compter les absents »). Documenter ce que mon outil fait *à* son utilisateur, pas seulement ce qu'il fait *pour* lui. Rendre mes choix réversibles tant que je n'ai pas la preuve qu'ils soignent (chapitre « Écouter sans obéir »). Et répondre de l'ensemble devant ceux qui en dépendent, dans une structure où ils ont voix — car le soin sans comptes à rendre redevient du pouvoir et de l'oppression (chapitre « Le pacte »).

L'éthique du soin s'adresse d'ailleurs cette mise en garde à elle-même : le soin n'a rien d'innocent — il peut opprimer, moraliser, décider du bien d'autrui à sa place — et María Puig de la Bellacasa en a fait une condition du concept plutôt qu'un embarras[9]. Ce serment n'échappe à l'objection que d'une seule façon : il n'engage que celui qui le prononce, et chacune de ses clauses se vérifie. On peut me prendre en défaut ; c'est le but.

Il reste à affronter les deux objections, et je prends la plus forte version de chacune. La première vient de l'intérieur même de l'éthique du care : en parlant de soin, ne reconduit-on pas le vieux partage — aux femmes la sollicitude, aux hommes la technique —, c'est-à-dire l'essentialisme que ces autrices ont précisément combattu ? L'objection a été faite à Gilligan et à Noddings, et la réponse est celle de Tronto : le soin n'est pas une

disposition féminine, c'est un travail, universellement requis, historiquement assigné aux subalternes et dévalorisé pour cette raison même[8]. Réclamer que l'ingénierie apprenne le soin, ce n'est pas féminiser un bastion : c'est exiger qu'un métier de privilégiés reprenne sa part d'un travail qu'il a toujours délégué — aux rédactrices de documentation, aux équipes de support, les forums d'entraide, tous les métiers invisibles du chapitre « Le travailleur invisible », et la sociologie de l'informatique montre qui, statistiquement, s'y retrouve. La seconde objection est celle de l'ingénieur : le soin, n'est-ce pas le paternalisme — simplifier de force, décider à la place de l'usager, l'infantiliser ? Elle mérite mieux qu'un haussement d'épaules, car un soin qui confisque est en effet une domination de plus, et le chapitre « Ouvrir sans tenir le pinceau » a montré des « simplifications » qui étaient des amputations. Mais l'objection confond le soin avec l'assistance. Le but du soin, en médecine comme en conception, est l'autonomie du soigné, et Illich a écrit le critère une fois pour toutes : « j'appelle société conviviale une société où l'outil moderne est au service de la personne intégrée à la collectivité, et non au service d'un corps de spécialistes. Conviviale est la société où l'homme contrôle l'outil »[24]. La profession du soin elle-même avait posé le critère avant lui : la définition fondatrice des soins infirmiers modernes — Virginia Henderson, 1966, apprise par chaque élève infirmière de France — assiste l'individu, malade ou bien portant, dans les actes « qu'il accomplirait par lui-même s'il avait assez de forces, de volonté ou de savoir », et le fait, dit la clause décisive, de façon à l'aider à « reconquérir son indépendance le plus rapidement possible »[25]. Un soin qui se mesure à la vitesse de sa propre disparition ne peut pas être le

paternalisme qu'on lui reproche. Prendre soin de l'utilisateur, ce n'est pas tenir le pinceau à sa place, c'est faire que l'outil soit à sa main, à sa portée, et sous son contrôle. Le paternalisme décide pour l'autre ; le soin rend l'autre capable de décider. Toute la différence du livre tient dans ce pli.

Il reste un dernier mot à désarmer, et il m'est adressé. Cette semaine encore, on me remerciait pour ma « vision » — le mot est dans le message, et il est d'abord de ma faute : la description de mon propre fork l'affiche, « moins le superflu, plus une vision de la conception », écrite un jour où les caractères étaient comptés[26]. Il faut pourtant refuser ce mot, et ce refus résume le livre. « Vision » est le récit du héros sous son autre nom : un don rare, logé dans un individu d'exception, inné donc intransmissible, admirable donc indiscutable — le registre exact que le chapitre « L'ambition non héroïque » a démonté chez les mainteneurs, et qu'on ne réinstallera pas au profit du concepteur. Ce qu'on appelle ma vision, ce livre vient de le passer en revue, pièce par pièce : c'est une méthode. Une part m'est venue de l'école d'ingénieurs, une part de l'intuition corrigée en chemin ; et tout ce qui en valait la peine s'est laissé réduire à des choses que la recherche étudie, à des questions qu'on s'impose de poser dans le bon ordre, à des priorités qu'on s'interdit de trahir — et, au bout, au soin de ceux qui ont besoin qu'on prenne soin d'eux. La médecine ne dit pas d'un bon clinicien qu'il a une vision ; elle dit qu'il a une discipline. Henderson a défini les soins infirmiers sans un mot pour le talent[25]. Il n'y a pas de visionnaires en conception : il y a des gens qui se sont imposé une discipline de soin — et des collectifs qui la leur ont rendue possible, ou

impossible. Ce qu'on croit un don se transmet ; ce qui se transmet s'enseigne ; et ce qui s'enseigne n'excuse plus personne.

Je peux maintenant dire pourquoi le livre devait se terminer ici. La question qui l'a ouvert était technique : qu'est-ce que concevoir ? La réponse, au bout du chemin, ne l'est plus. Concevoir, c'est prendre soin d'autrui à travers un objet technique — c'est la forme que prend le soin quand il doit passer par la machine. Un monde qui se technicise sans demander l'avis de personne fabrique chaque jour des êtres technicisés, qui n'ont pas choisi de l'être et qui doivent vivre quand même. Quelqu'un doit les aider à vivre là, comme la médecine aide à vivre tout court. C'est nous. Ce métier n'est pas un droit à exercer sur les autres ; c'est une charge à porter pour eux. Le logiciel libre a passé 30 ans à libérer le code. La chose qui reste à libérer, c'est l'utilisateur — et on ne libère pas quelqu'un qu'on méprise.

---

1. Oldenziel, R. (1999). *Making Technology Masculine : Men, Women, and Modern Machines in America, 1870-1945*. Amsterdam : Amsterdam University Press. https://doi.org/10.5117/9789053563816 . Champ d'étude : les États-Unis, 1870-1945 — la généralisation au métier entier est mon extrapolation, que les chiffres contemporains de la note suivante soutiennent. ↩

2. American Society for Engineering Education (2024). *Engineering & Engineering Technology by the Numbers, 2023*. Washington, DC : ASEE https://ira.asee.org/by-the-numbers/ , tables 1 et 4 — femmes : 24,6 % des diplômes de

licence d'ingénierie ; génie mécanique : première discipline en volume (29 792 diplômes) et 17,9 % de femmes ; l'aérospatiale (17,4 %), le pétrole (16,4 %), l'informatique (14,9 %) et l'électricité (14,6 %) font moins encore. Données américaines ; les proportions varient d'un ou deux points selon les éditions. ↩

3. 1,5 % : enquête développeurs FLOSS 2002 (Ghosh, R. A., Glott, R., Krieger, B. et Robles, G., *Free/Libre and Open Source Software : Survey and Study*, International Institute of Infonomics, Univ. de Maastricht, 2002) https://www.math.unipd.it/~bellio/FLOSS%20Final%20Report%20-%20Part%204%20-%20Survey%20of%20Developers.pdf, chiffre repris et discuté — contre 28 % dans le logiciel propriétaire (NSF, 2004) — par le rapport genre du projet FLOSSPOLS : Nafus, D., Leach, J. et Krieger, B. (2006). « Gender : Integrated Report of Findings », livrable D16, Univ. de Cambridge / Commission européenne, p. 4. https://www.jamesleach.net/downloads/FLOSSPOLS-D16-Gender_Integrated_Report_of_Findings.pdf 3 % : GitHub, *Open Source Survey 2017* https://opensourcesurvey.org/2017/ (« 95 % of respondents are men ; just 3 % are women and 1 % are non-binary » — notre traduction dans le corps du texte). 8,1 % des commits publics (13,5 % des auteurs au genre inférable) : Rossi, D. et Zacchiroli, S. (2022). « Worldwide Gender Differences in Public Code Contributions… », *ICSE-SEIS'22*, ACM https://doi.org/10.1145/3510458.3513011 — inférence par prénom, couvrant 64 % des auteurs, biais culturels reconnus par les auteurs. 14 % : Linux Foundation Research (2021), *Diversity, Equity, and Inclusion in Open*

Source https://www.linuxfoundation.org/research/the-2021-linux-foundation-report-on-diversity-equity-and-inclusion-in-open-source — enquête auto-déclarative. Aucun chiffre unique n'est honnête sans sa méthode ; la fourchette, elle, est stable dans sa laideur. ↩

4. Gilligan, C. (1982). *In a Different Voice : Psychological Theory and Women's Development*. Cambridge, MA : Harvard University Press https://doi.org/10.2307/j.ctvjk2wr9 ; trad. fr. *Une voix différente. Pour une éthique du care*, trad. A. Kwiatek révisée par V. Nurock, présentation de S. Laugier et P. Paperman, Paris : Flammarion, « Champs Essais », 2008 (première trad. fr. : *Une si grande différence*, Flammarion, 1986). ↩

5. Noddings, N. (1984). *Caring : A Feminine Approach to Ethics and Moral Education*. Berkeley : University of California Press https://books.google.com/books/about/Caring.html?id=vkMkLi6pnMYC — les deux états qu'elle décrit, *engrossment* (attention absorbée par l'autre) et *motivational displacement* (déplacement de la motivation vers les besoins de l'autre), sont donnés ici dans nos termes : l'ouvrage n'a pas de traduction française établie. ↩

6. Held, V. (2006). *The Ethics of Care : Personal, Political, and Global*. New York : Oxford University Press, p. 17 https://doi.org/10.1093/0195180992.001.0001 : « There can be no justice without care […] for without care no child would survive and there would be no persons to respect » (notre traduction dans le corps du texte). Held (1929-2026) tenait le

soin pour « probablement la valeur la plus fondamentale » (même page). ↩

7. Fisher, B. et Tronto, J. C. (1990). « Toward a Feminist Theory of Caring », dans E. K. Abel et M. K. Nelson (dir.), *Circles of Care*. Albany : SUNY Press, p. 40 https://sunypress.edu/Books/C/Circles-of-Care ; repris dans Tronto, J. C. (1993). *Moral Boundaries : A Political Argument for an Ethic of Care*. New York : Routledge, p. 103. https://doi.org/10.4324/9781003070672 La citation française suit la traduction publiée : *Un monde vulnérable. Pour une politique du care*, trad. H. Maury, Paris : La Découverte, 2009, p. 143 ; la fin de la phrase (« de sorte que… ») est ici paraphrasée hors guillemets, la traduction publiée n'ayant été collationnée que sur le premier membre. ↩

8. Tronto, *Moral Boundaries*, *op. cit.*, p. 105-108 (quatre phases, et leurs qualités morales : attention, responsabilité, compétence, réactivité) ; la cinquième phase, *caring with*, est ajoutée dans Tronto, J. C. (2013). *Caring Democracy : Markets, Equality, and Justice*. New York : NYU Press https://doi.org/10.18574/nyu/9780814770450.001.0001 — qui porte aussi l'argument anti-essentialiste : le soin comme travail dévalorisé et assigné, non comme disposition de genre. ↩ ↩

9. Puig de la Bellacasa, M. (2011). « Matters of Care in Technoscience : Assembling Neglected Things ». *Social Studies of Science*, 41(1), 85-106 https://doi.org/10.1177/0306312710380301 ; et (2017) *Matters of Care : Speculative Ethics in More Than Human Worlds*.

Minneapolis : University of Minnesota Press. https://doi.org/10.5749/9781452953465 L'agenda IHM : Bardzell, S. (2010). « Feminist HCI : Taking Stock and Outlining an Agenda for Design ». *CHI '10*, ACM, p. 1301-1310 https://doi.org/10.1145/1753326.1753521 — six qualités : pluralisme, participation, plaidoyer, écologie, incarnation, auto-divulgation (nos traductions). ↩ ↩

10. Jones, P. H. (2013). *Design for Care : Innovating Healthcare Experience*. Brooklyn, NY : Rosenfeld Media https://rosenfeldmedia.com/books/design-for-care/ , chap. 1, « Design as Caregiving » — le cadre est explicitement celui des systèmes de santé ; l'extension du soin à toute conception est ma thèse, pas la sienne. ↩

11. Ellul, J. (1954). *La Technique ou l'Enjeu du siècle*. Paris : Armand Colin https://www.sudoc.fr/013436104 — l'auto-accroissement et l'autonomie de la technique, qui s'impose comme milieu sans délibération, sont le cœur du livre ; énoncé ici en discours indirect. ↩

12. Winner, L. (1986). *The Whale and the Reactor : A Search for Limits in an Age of High Technology*. Chicago : University of Chicago Press https://doi.org/10.7208/chicago/9780226902098.001.0001 , chap. 1, p. 10 : « we so willingly sleepwalk through the process of reconstituting the conditions of human existence » (notre traduction dans le corps du texte) — le « somnambulisme technologique ». ↩

13. Noble, D. F. (1995). *Progress Without People : New Technology, Unemployment, and the Message of Resistance*. Toronto : Between the Lines https://archive.org/details/progresswithoutpoooonobl (1re éd. : *In Defense of Luddism*, Chicago : Charles H. Kerr, 1993) — en discours indirect. ↩

14. Feenberg, A. (1999). *Questioning Technology*. Londres : Routledge. https://doi.org/10.4324/9780203022313 Le concept apparaît d'abord sous le nom de « subversive rationalization » : Feenberg, A. (1992). « Subversive Rationalization : Technology, Power, and Democracy ». *Inquiry*, 35(3-4), 301-322 https://doi.org/10.1080/00201749208602296 ; il est rebaptisé « democratic rationalization » dans le livre de 1999. ↩

15. Van Dijk, J. (2005). *The Deepening Divide : Inequality in the Information Society*. Thousand Oaks : Sage https://doi.org/10.4135/9781452229812 — l'accès comme chaîne de barrières (motivation, matériel, compétences, usages) ; Hargittai, E. (2002). « Second-Level Digital Divide : Differences in People's Online Skills ». *First Monday*, 7(4) https://doi.org/10.5210/fm.v7i4.942 ; Wyatt, S. (2003). « Non-Users Also Matter », dans N. Oudshoorn et T. Pinch (dir.), *How Users Matter*. Cambridge, MA : MIT Press, p. 67-79 https://doi.org/10.7551/mitpress/3592.003.0006 — contre la lecture du non-usage comme déficit individuel. ↩

16. Norman, *The Design of Everyday Things*, *op. cit.*, chap. 2, sections « Falsely Blaming Yourself » et « Learned

Helplessness » (éd. 1988, p. 39-40) : l'auto-accusation de l'utilisateur et l'impuissance apprise. ↩

17. Steinmacher, I., Conte, T., Gerosa, M. A. et Redmiles, D. (2015). « Social Barriers Faced by Newcomers Placing Their First Contribution in Open Source Software Projects ». *CSCW '15*, ACM, p. 1379-1392 https://doi.org/10.1145/2675133.2675215 — 58 barrières recensées, dont treize sociales (accueil hostile, réponses méprisantes ou absentes) ; en complément sur le registre verbal du milieu : Squire, M. et Gazda, R. (2015). « FLOSS as a Source for Profanity and Insults ». *HICSS-48*, IEEE, p. 5290-5298. https://doi.org/10.1109/hicss.2015.623 L'expression « lis donc le manuel » est ma paraphrase polie d'un sigle que ces études n'emploient pas. ↩

18. Cassell, E. J. (1982). « The Nature of Suffering and the Goals of Medicine ». *New England Journal of Medicine*, 306(11), p. 639 https://doi.org/10.1056/NEJM198203183061104 : « The relief of suffering and the cure of disease must be seen as twin obligations of a medical profession that is truly dedicated to the care of the sick » (notre traduction dans le corps du texte) ; développé dans Cassell, E. J. (1991). *The Nature of Suffering and the Goals of Medicine*. New York : Oxford University Press. https://doi.org/10.1093/acprof:oso/9780195156164.001.0001 ↩

19. Stiegler, B. (2008). *Prendre soin. Tome 1 : De la jeunesse et des générations*. Paris : Flammarion https://www.cultura.com/p-prendre-soin-de-la-jeunesse-et-des-generations-

9782081207363.html ; et (2010) *Ce qui fait que la vie vaut la peine d'être vécue. De la pharmacologie*. Paris : Flammarion https://editions.flammarion.com/ce-qui-fait-que-la-vie-vaut-la-peine-detre-vecue/9782081220355 — la citation est celle de la présentation de l'éditeur du volume de 2010, vérifiée sur pièce ; le titre du présent chapitre assume l'écho. ↩

20. Hippocrate, *Épidémies*, livre I, 2e section, § 5, trad. É. Littré, *Œuvres complètes d'Hippocrate*, t. II, Paris : J.-B. Baillière, 1839-1861. https://archive.org/details/BIUSante_34859x02 Contrairement au folklore, la formule latine *primum non nocere* ne figure pas dans le Serment ; sur son origine réelle, voir Smith, C. M. (2005). « Origin and Uses of Primum Non Nocere ». *The Journal of Clinical Pharmacology*, 45(4), 371-377. https://doi.org/10.1177/0091270004273680 ↩

21. Papanek, V. (1971). *Design for the Real World : Human Ecology and Social Change*. New York : Pantheon Books https://books.google.com/books/about/Design_for_the_Real_World.html?id=gf5TAAAAMAAJ ; trad. fr. *Design pour un monde réel*, trad. R. Louit et N. Josset, Paris : Mercure de France, 1974. L'ouverture, ici paraphrasée : « There are professions more harmful than industrial design, but only a very few of them » (préface ; la pagination varie selon les éditions). ↩

22. Monteiro, M. (2019). *Ruined by Design : How Designers Destroyed the World, and What We Can Do to Fix It*. San Francisco : Mule Books https://www.ruinedby.design/ ; et (2017) « A Designer's Code of Ethics », Dear Design Student https://deardesignstudent.com/a-designers-code-of-ethics-

f4a88aca9e95. L'attribution est à prendre au mot près : Monteiro plaide pour un code d'éthique et une responsabilité professionnelle organisée sur le modèle médical ; l'expression « serment d'Hippocrate du designer » vient de ses commentateurs, pas de ses textes. ↩

23. Ehmke, C. A. (2019). *The Hippocratic License* https://firstdonoharm.dev/ (versions ultérieures : HL3, Organization for Ethical Source). Le rejet s'appuie sur l'*Open Source Definition* (critères 5 et 6 : non-discrimination de personnes et de domaines d'activité, OSI) https://opensource.org/osd et, côté FSF, sur la « liberté 0 » d'exécuter le programme pour n'importe quel usage. https://www.gnu.org/philosophy/free-sw.html ↩

24. Illich, I. (1973). *La convivialité*. Paris : Seuil https://catalogue.bnf.fr/ark:/12148/cb43897048t — version française établie par l'auteur avec L. Giard et V. Bardet (éd. originale : *Tools for Conviviality*, New York : Harper & Row, 1973) ; citation de l'introduction, vérifiée sur le texte français. ↩

25. Henderson, V. (1966). *The Nature of Nursing*. New York : Macmillan ; trad. fr. *La nature des soins infirmiers*, Paris : InterÉditions, 1994. La définition, d'abord publiée en 1955 (Harmer & Henderson, *Textbook of the Principles and Practice of Nursing*, 5e éd., Macmillan), fut diffusée mondialement par le Conseil international des infirmières (*Basic Principles of Nursing Care*, Genève, 1960, traduit en une trentaine de langues) ; le rendu français cité est celui de l'enseignement

infirmier, où la grille des 14 besoins fondamentaux de Henderson structure toujours le recueil de données. ↩ ↩

26. Message reçu le 31 juillet 2026 sur le dépôt d'Ansel : « Thanks to Aurelien for your vision, patience and perseverance, and to Jiyone for taking over and keeping the project going ! » (discussion #812 https://github.com/aurelienpierreeng/ansel/discussions/812#discussioncomment-17853594). La description du dépôt, de ma main : « A Darktable fork minus the bloat plus some design vision. » ↩

Épilogue — Derrière chaque problème, un humain

Il aura fallu l'écriture de ce livre pour mettre des mots et des concepts, définis hors de moi, sur le fiasco de mon expérience avec Darktable. La conclusion tient en peu de mots : des objectifs, des valeurs et des priorités irréconciliables. Et j'ajoute sans détour que mes objectifs et mes priorités sont meilleurs que les leurs, en matière d'impact social hors du cercle fermé. Ce n'est pas une affaire de personnes. Mes priorités se mesurent à ceux qui ne sont pas dans la pièce : l'utilisateur à qui on a imposé le numérique sans le lui demander, et qui n'a aucun moyen de se défendre contre un mauvais outil. Les leurs se mesurent à eux-mêmes — au plaisir de coder, à la technique qu'on apprend, à l'entre-soi qu'on garde. Servir qui ne peut pas se défendre plutôt que soi-même, ce n'est pas une vertu que je m'attribue : c'est la définition d'un bénéfice social. Et la mienne se vérifie — mes enquêtes d'utilisateurs sont publiées avec leurs données — quand la leur se réclame d'une « émancipation » que le dossier dément. Un objectif qu'on mesure et qu'on assume vaut mieux qu'un objectif qui se cache derrière un slogan.

On délègue au club informatique du lycée la conception et la fabrication d'outils destinés à des gens qui ne sont pas dans la pièce : des gens qui s'amusent avec des ordinateurs, souhaitent apprendre de nouvelles techniques, et cherchent un alibi pour le

faire. Cet amateurisme est structurel : c'est ainsi que sont recrutés les contributeurs, exonérés de toute responsabilité par la licence libre et le caractère bénévole de leur travail. Il n'est ni bon ni mauvais, juste déplacé si l'on vise à apporter un bénéfice social dans un contexte, je le répète, où l'on a imposé les technologies numériques aux gens sans rien leur demander, sur une échelle de temps très courte.

Ce qui est coupable, c'est le double discours — celui que le prologue annonçait, et que les chapitres ont depuis mesuré sur pièces : la façade qui prétend à l'émancipation de l'utilisateur, à l'inclusivité sans condition, à la maturité des outils en contexte d'utilisation professionnelle, le « par des photographes pour des photographes » auquel je me suis laissé prendre, alors que la réalité montre de grands enfants qui jouent avec le code, au mépris des utilisateurs, et avec une absence de rigueur et de méthode qu'ils n'oseraient probablement pas se permettre dans leur « vrai métier » (c'est une hypothèse charitable de ma part, en fait je n'en sais rien).

Cette façon de créer du logiciel libre, sans méthode de conception, sans structure pour la garantir, sans comptes à rendre associés à des responsabilités nommées et bornées, avec une fausse horizontalité qui dissimule le pouvoir réel de certains, en appliquant des échelles de valeurs et des cultes du cargo que la littérature scientifique démonte, ne sert personne : elle ne crée que de l'entropie. On s'agite, on s'enthousiasme, mais le travail passé ne fait qu'augmenter la facture de travail futur, une facture que quelqu'un d'autre payera, et les utilisateurs sont les grands oubliés du système. On préfère alors créer un énième nouvel

éditeur d'images, à présent entièrement écrit par IA grâce aux LLM, et donc à la portée de presque n'importe qui, et accumuler les produits à moitié finis. Vous n'avez qu'à choisir le vôtre — illusion de choix : un vrai choix se ferait entre des outils qui marchent ; ici, on ne choisit que les fonctionnalités dont on se passera.

Ce système est aveugle à lui-même : l'absence de ce qu'il considère comme des gros mots (méthode, structure, responsabilités, organisation, objectifs et métriques falsifiables, compétences) est ce qui lui coûte le plus en temps et en travail, mais également la source de violences et d'injustices systémiques, car tout se règle par confrontations de subjectivités, faute de critères objectifs et falsifiables, et, de toute façon, l'utilisateur final n'est même pas consulté. Tous ceux qui osent pointer les dérives, les abus, les incohérences et les paradoxes sont punis par la paroisse. Le système se reproduit à force de n'attirer (ou de ne garder) que les mêmes profils que ceux qui ont créé le problème. Tout ça, encore une fois, en faisant des promesses d'émancipation qui n'engagent que ceux qui y croient.

Mais tout ça n'est que les ramifications visibles du problème de fond, et patcher le problème sur son volet social (voire sociétal) comme on patche le code, par petits hacks unitaires, ne réglera rien ni sur le fond ni sur la forme. Le logiciel libre est à l'outillage informatique ce que l'hôpital public est à la clinique privée : le service public de tout le monde, sans distinction de quoi que ce soit. Pas le terrain de jeu public mais finalement privé d'une minorité de programmeurs oisifs qui peuvent jouer à l'ordinateur entre 17 h et 21 h pendant que le repas se prépare « tout seul ».

Dans et hors de l'open-source, la conception bien faite appartient au domaine du *care*/soin. C'est la conclusion vers laquelle je cheminais depuis des années — bien avant que l'écriture de ce livre ne lui donne son nom —, à force de chercher pourquoi je me reconnaissais de moins en moins dans le logiciel libre et son éthique (réelle, pas celle affichée et fantasmée). Le premier programme informatique que j'ai écrit, en 2012, s'appelait Scan2Thunderbird, et je l'ai écrit pour mon grand-père qui, ayant commencé l'informatique vers 72 ans, n'arrivait pas à envoyer de pièce jointe numérisée dans un email. L'application est un simple script qui ouvre l'un après l'autre les logiciels requis (celui pour scanner, page par page, puis celui pour envoyer), avec un ensemble réduit d'options utilisateur (limité aux cas courants), et crée un nouvel email où le document numérisé est déjà joint. Il ne reste alors plus qu'à rédiger (à deux doigts). Pas besoin de se souvenir quoi faire, quel logiciel utiliser (quel est son nom et où il est dans le menu), ni dans quel ordre : on est guidé, étape par étape, par une interface unique. Voilà de la conception qui réduit les frictions et allège la douleur, au service de quelqu'un qui n'aurait pas pensé à la demander.

Ce n'était pas une affaire de famille : mes meilleures contributions au logiciel libre ont toutes eu cette forme-là, et aucune n'était un exploit technique. Trois exemples, tirés du même dossier Darktable.

Pendant des années, sur le forum français de Darktable, la même question est revenue : comment agrandir la police de l'interface, trop petite sur les écrans récents. Et pendant des années, la même réponse est tombée, du mainteneur comme des habitués : quitter

le logiciel, retrouver un fichier de configuration caché, l'ouvrir dans un éditeur de texte, y changer à la main `screen_dpi_overwrite` — sans se tromper de signe[1]. Personne n'y voyait de problème : on répondait à la question, encore, au lieu de la supprimer. Je m'y suis usé. J'ai fini par refondre l'interface pour que tout — panneaux, boutons, marges — suive la taille de police et la résolution de l'écran ; le travail était brouillon, c'était l'une de mes premières contributions en C, et le mainteneur a dû le reprendre pour le fusionner[2]. L'année suivante, le réglage est arrivé dans les préférences, à un clic, et la question s'est éteinte sur le forum. La solution était simple, l'impact énorme, et personne n'y avait pensé : répondre dix fois à la même question passait pour de l'entraide, quand c'est un défaut de conception.

En novembre 2021, sur le même forum, un utilisateur daltonien est venu poser son problème : un retoucheur qui ne distingue pas certaines couleurs peut faire passer un visage du rouge au vert sans s'en apercevoir, et il n'a aucun moyen de le vérifier seul — il lui faut demander à un humain valide : « ici, quelle est la couleur ? et là ? ». Il ne réclamait pas un miracle ; son fil s'intitulait « Gardes de fou ou cannes pour daltonien ? » : que la pipette dise les couleurs avec des mots. Deux mois plus tard, j'ai livré un vocabulaire de couleurs — une table qui relie les coordonnées de chromaticité à des noms, complétée des teintes moyennes de peau mesurées sur six populations — affiché dans la pipette d'échantillonnage[3]. C'est à ce jour la seule tentative d'accessibilité dans ce logiciel, et elle est née de la demande d'un seul utilisateur. Ce n'est pas un mauvais ratio, c'est le bon : une

personne réelle, nommée, située vaut tous les « utilisateurs moyens » élastiques des discussions de projet.

Le reste du champ donne la mesure de ce que ce soin a d'exceptionnel — c'est-à-dire d'anormal. À ce jour, aucun autre éditeur photo libre — RawTherapee, ART, Filmulator, PhotoFlow, digiKam — n'a aidé un daltonien à vérifier les couleurs de ses images. Le mieux qu'on y trouve est la sympathie sans suite : chez RawTherapee, en 2020, un utilisateur daltonien demande à pouvoir seulement distinguer l'aperçu des masques ; « voilà une demande sur laquelle nous devrions travailler », répond l'équipe — le ticket est toujours ouvert six ans plus tard, avec un unique commentaire et zéro ligne de code. Le pire est plus franc : en 2014, sur le même projet, un contributeur répond au mainteneur qui proposait des pictogrammes accessibles : « je suis un peu perplexe : pourquoi un daltonien aurait-il besoin de développer des photos ? ». Quant aux filtres « daltonisme » de GIMP ou de G'MIC, ils font l'inverse du soin : ils simulent l'infirmité au bénéfice des valides, pour leur permettre de vérifier leurs designs. Personne n'outille le daltonien qui retouche, parce que personne n'y pense — il n'est pas dans la pièce[4]. L'argument n'est pas technique : les 413 lignes de code que j'ai mis dans Darktable pour cette finalité en 2022 sont réutilisables pratiquement telles quelles par tout éditeur d'image écrit en C ou en C++. Et tous le sont : RawTherapee, ART, Filmulator, PhotoFlow, digiKam, comme GIMP, Krita, Siril ou vkdt. Rien à inventer, rien à financer : tout à copier ; l'open-source en est parfaitement capable, quand ça lui chante, il le montre régulièrement. S'il ne

s'est rien passé, c'est une pure question de priorités : le problème est politique.

Restent les ordres de grandeur : ces logiciels comptent leurs utilisateurs en millions. GIMP revendique cinquante mille téléchargements par jour ; Krita en a compté 5,6 millions pour la seule année 2021 ; Darktable a servi 8,5 millions de téléchargements de binaires, dont 213 000 pour la dernière version stable en cinq semaines[5]. Un public presque exclusivement masculin — 98 % chez Darktable, seule mesure disponible — où le daltonisme rouge-vert touche 8 % des hommes : environ un utilisateur sur treize. Sur des millions, cela fait des dizaines de milliers de daltoniens laissés sans outil, au bas mot. Voilà combien de personnes ne sont pas dans la pièce.

Fin 2019, enfin, j'avais ouvert un ticket chez Darktable pour demander des paquets compilés automatiquement chaque nuit : que ceux qui ne savent pas compiler puissent tester la version de développement, signaler les régressions avant la sortie plutôt qu'après, ou simplement s'en servir. Le mainteneur a trouvé l'idée bonne. Puis rien : le ticket a pourri trois ans et demi sous le robot de péremption, la doctrine restant « compilez vous-même, ce n'est pas difficile ». Et la doctrine se payait au guichet : à chaque sortie, des utilisateurs venaient réclamer sur le traqueur la mise à jour de dépôts tiers en retard, et on leur répondait que le projet ne maintient pas de paquets[6]. En décembre 2022, j'ai monté ces compilations nocturnes pour Ansel — 70 heures de travail, pour un ingénieur mécanique de formation qui a dû apprendre l'empaquetage Linux et Windows en le faisant[7]. Deux problèmes fermés d'un coup : les testeurs sans compilateur ont eu

leurs paquets du jour, et les dépôts tiers ont cessé d'être le problème de personne. Darktable a fini par publier sa propre version nocturne — six mois après la mienne, trois ans et demi après ma demande.

Trois fois la même forme : une douleur réputée triviale par ceux qui ne la vivent pas, une solution simple que personne n'avait prise au sérieux, un effet durable. Le soin n'est pas une vertu du caractère, c'est une méthode de conception : elle commence par tenir la question d'en face pour digne d'être supprimée — pas indéfiniment répondue.

Le logiciel libre ne peut pas rester l'amusement de quelques-uns qui peuvent, au détriment de la majorité qui a besoin. Il faut changer, d'abord de vision, et ensuite de structure.

Ça ne pourra se faire qu'en changeant le personnel, parce qu'on ne peut pas attendre de ceux qui ont créé le problème qu'ils le résolvent : pour eux, il n'y a pas de problème. En abolissant la propriété intellectuelle et industrielle sur le code source des logiciels, ils ont libéré l'utilisateur expert de la mainmise des fabriquants et lui ont rendu le contrôle de sa machine. Et en ayant libéré l'utilisateur expert, ils pensent avoir libéré tout le monde. C'est qu'ils ne s'intéressent qu'à eux, et à ceux qui leur ressemblent. Ce ne sont pas quelques heures de formation et un nouveau code de conduite qui vont transformer le club informatique en hôpital public de la technicisation : ils ne sont pas là pour ça, et n'y ont aucun intérêt.

Ce qu'il faut faire, c'est laisser le logiciel libre à ses geeks, et inventer le logiciel public : avec le code libre, mais à l'abri des libristes. On ne peut pas demander à des privilégiés de penser l'émancipation de l'Autre sans tomber, au mieux, dans la condescendance.

---

1. Fils du forum Darktable.fr : « Taille de l'interface » https://Darktable.fr/t/taille-de-linterface/206 (2016), « Police des outils » https://Darktable.fr/t/police-des-outils/831 (2017-2018), « Taille des caractères » https://Darktable.fr/t/taille-des-caracteres/2005 (février 2019), « Interface trop grande :(» https://Darktable.fr/t/interface-trop-grande/2489 (2019-2020). Dans le deuxième, le mainteneur Pascal Obry donne la marche à suivre — éditer `~/.config/Darktable/Darktablerc` et changer `screen_dpi_overwrite` — en précisant qu'il faut « quitter Darktable AVANT de faire le changement car Darktablerc est réécrit lorsque Darktable se termine » ; l'utilisateur venait de se tromper de signe sur la valeur. En février 2019, un habitué constate : « il y a déjà eu plusieurs réponses sur ce sujet dans le forum ». ↩

2. La refonte : PR #2037 https://github.com/Darktable-org/Darktable/pull/2037 (janvier 2019, 132 commentaires), close sans fusion — Pascal Obry la reprend, la corrige et la fusionne lui-même dans la PR #2381 https://github.com/Darktable-org/Darktable/pull/2381 (Darktable 3.0) : « Merged in #2381 with a some fixes

(redraw, accels). Thanks again for the hard work on this quite delicate issue Aurélien. » Le réglage en préférences arrive avec Darktable 3.2 : le champ de taille de police par Chris Elston (PR #4974 https://github.com/Darktable-org/Darktable/pull/4974), l'exposition du DPI et des facteurs d'échelle par moi (PR #5469 https://github.com/Darktable-org/Darktable/pull/5469) — « it's long due and has been requested many many times », écrivais-je alors. ↩

3. Fil « Gardes de fou ou cannes pour daltonien ? » https://Darktable.fr/t/gardes-de-fou-ou-cannes-pour-daltonien/3834, Darktable.fr, 12 novembre 2021. L'auteur, trichromate deutan : « j'ai besoin d'une aide humaine pour m'assurer que le résultat n'a pas trahi mon objectif initial : "ici, quelle est la couleur ? et là ?" ». Le code : commit du 20 janvier 2022 https://github.com/Darktable-org/Darktable/commit/dfe62a62211aa160420ca1c711749764ecfd7894 et PR #10935 https://github.com/Darktable-org/Darktable/pull/10935, fusionnée manuellement par Pascal Obry après remise en forme ; livré dans Darktable 4.0 https://www.Darktable.org/2022/07/Darktable-4.0.0-released/ (juillet 2022) sous le nom de « color glossary » : noms de couleurs dans l'infobulle de la pipette globale, et teintes de peau moyennes pour trois zones du corps et six populations, d'après Xiao et al. (2017) et De Rigal et al. (2010). ↩

4. Audit du 29 juillet 2026 sur les traqueurs, manuels et notes de version de RawTherapee, ART, Filmulator, PhotoFlow, digiKam, GIMP, Krita, LightZone, vkdt, PhotoFlare, Fotoxx

et Siril. RawTherapee 2020 : demande sur le fil Local adjustments https://discuss.pixls.us/t/local-adjustments-is-in-dev/18795/24 — réponse d'un développeur : « Imho this is a request we should work on ! » — puis ticket #5807 https://github.com/RawTherapee/RawTherapee/issues/5807, ouvert le 21 juin 2020, toujours ouvert, un seul commentaire. RawTherapee 2014 : ticket #733 https://github.com/RawTherapee/RawTherapee/issues/733 — « I am a bit puzzled why would cor blind person would need to do raw imaging » (sic), en réponse au mainteneur qui proposait d'adopter les pictogrammes ColorADD. digiKam : aucun signalement lié au daltonisme jamais déposé sur bugs.kde.org. GIMP : filtre d'affichage « Color Deficient Vision » https://docs.gimp.org/2.10/en/gimp-display-filter-dialog.html, simulation destinée aux voyants ; G'MIC : commande `colorblind` « to check the accessibility of a web site or a graphical document ». Seule nuance du champ : Siril (empileur astrophotographique) a rendu configurables les couleurs de ses annotations « For color-blind people » (2023) — un geste d'interface, rien sur les couleurs de l'image — tout en laissant sans aucune réponse la demande directe d'un utilisateur daltonien https://discuss.pixls.us/t/change-colour-of-background-extraction-points-for-colour-blind-users/36352 sur son forum. En face, les daltoniens qui se servent du glossaire de Darktable en témoignent, en 2024 https://discuss.pixls.us/t/Darktable-color-picker-color-names/45604 comme en 2026 https://discuss.pixls.us/t/hue-value-to-color-convert/56595 : « It's like being without leg for years and you mounted me a new fresh prosthesis. »

Langages vérifiés sur les statistiques de dépôt (29 juillet 2026) : RawTherapee, ART, Filmulator, PhotoFlow, digiKam, Krita et PhotoFlare sont écrits en C++ ; GIMP, Siril et vkdt en C ; G'MIC en C++. LightZone est une application Java, mais son décodage RAW et une partie de son traitement vivent dans un cœur natif C/C++ (statistiques de dépôt : davantage de C que de Java) — et porter une table de correspondance de 413 lignes se fait dans n'importe quel langage. ↩

5. GIMP : « GIMP is downloaded over fifty thousand times a day and is used massively by millions of people across the world » (https://www.gimp.org/docs/userfaq.html, consultée le 29 juillet 2026), plus 3,7 millions d'installations sur le seul canal Flathub (Linux). Krita : 5 637 579 téléchargements en 2021 https://krita.org/en/posts/2022/krita-in-2021-and-2022/, dernier chiffre annuel publié — le projet a cessé de publier ses statistiques en 2023. Darktable : 8 550 205 téléchargements de binaires cumulés sur GitHub (API, 29 juillet 2026), dont 212 715 pour la version 5.6.0 dans ses cinq premières semaines. Ni RawTherapee, ni digiKam, ni les autres ne publient de chiffres ; les téléchargements ne sont pas des utilisateurs, mais ces planchers, mesurés canal par canal, suffisent à l'ordre de grandeur. Le 98 % : 25 femmes sur 1 243 réponses, enquête Darktable 2022 https://eng.aurelienpierre.com/2026/07/who-are-the-Darktable-users-in-2022/ (n = 1 253). La prévalence : environ 8 % des hommes et 0,4 % des femmes d'ascendance européenne, 4 à 6,5 % des hommes en Asie de l'Est — J. Birch, « Worldwide prevalence of red-green color

deficiency » https://doi.org/10.1364/JOSAA.29.000313, *Journal of the Optical Society of America A*, 29(3), 2012. Le calcul : 0,98 × 8 % + 0,02 × 0,4 % ≈ 7,9 %, soit un utilisateur sur treize. ↩

6. Par exemple #5731 https://github.com/Darktable-org/Darktable/issues/5731 (2020) — réponse : « The Darktable developers do not maintain distribution packages » — et #11780 https://github.com/Darktable-org/Darktable/issues/11780 (2022) : « The Darktable project does not maintain the OBS repo. It is maintained by the community and updated when the community has time to do so. » ↩

7. La demande : #3925 « Provide an appimage ? » https://github.com/Darktable-org/Darktable/issues/3925, 29 décembre 2019 — accueil favorable du mainteneur (« Would be nice indeed ! »), puis étiquette du robot de péremption, et fermeture le 10 juin 2023 : « We now have an AppImage in the nightly build. » La mise en place chez Ansel : pré-version continue v0.0.0 https://github.com/aurelienpierreeng/ansel/releases/tag/v0.0.0 publiée le 13 décembre 2022, chaîne AppImage Linux le 14 https://github.com/aurelienpierreeng/ansel/commit/3852b1ba2c0be84c5130d4c21844f3199b3624b2. Les 70 heures : journal de développement Ansel, mars 2023 https://ansel.photos/en/news/dev-diary-1/ — « I spent a total 70 h on making the nightly packages builds for Windows and Linux work for continuous delivery », « something that

Darktable never got right (“you can build yourself, it’s not difficult”) ». ↩

Annexe — Les instruments de mesure

Ce livre s'est engagé, dès le prologue, à être falsifiable. Un chiffre qu'on ne peut pas re-mesurer n'est pas une mesure : c'est un argument d'autorité avec des décimales. Cette annexe tient l'engagement. Elle publie les instruments — les scripts qui produisent chaque série mesurée du livre — pour que n'importe qui, avec un ordinateur et une heure devant lui, puisse refaire les comptes, et me contredire chiffres en main. Le chapitre « Mesurer sans se mentir » posait la règle ; voici la paillasse.

Trois précisions honnêtes avant les scripts. D'abord, les mesures du livre sont des photographies datées — la plupart de juillet-août 2026, la date exacte figure dans chaque note. Re-mesurer aujourd'hui donnera des valeurs légèrement différentes, parce que le projet vit : entre le 1er et le 4 août 2026, le nombre de propositions fusionnées en 2025 renvoyé par l'API GitHub est passé de 864 à 866. Deux propositions requalifiées en trois jours ne changent aucune conclusion ; c'est l'ordre de grandeur du bruit, et il est dit. Ensuite, ces scripts sont des reconstructions au propre : les mesures d'origine ont été faites avec des commandes équivalentes, au fil de l'enquête, puis leurs méthodes ont été documentées pièce par pièce ; les scripts ci-dessous réimplémentent ces méthodes documentées, et j'ai vérifié qu'ils reproduisent les séries publiées partout où une vérification locale était possible. Enfin, une source n'est pas publique : l'export du salon Matrix/IRC. Le script de comptage est publié, l'export ne

l'est pas — les notes du chapitre « Le cercle fermé » en décrivent la provenance et les limites à chaque usage.

Il faut Python 3, le module `requests`, `git`, et pour les scripts GitHub un jeton personnel dans la variable d'environnement `GITHUB_TOKEN` (l'API GraphQL l'exige ; le quota gratuit suffit largement). Un clone local du dépôt étudié est nécessaire pour les deux premiers scripts — un clone sans blobs suffit et pèse dix fois moins : `git clone --filter=blob:none https://github.com/Darktable-org/Darktable`. Les commentaires des scripts sont en anglais, langue du code.

Le dépôt : démographie et rythme

Ce script produit les séries des chapitres « Livrer, pas larguer » et « Le comité n'inventera rien » : contributeurs actifs par an et par palier d'engagement (≥ 1, ≥ 5, ≥ 15 commits), commits médians du cœur (les ciseaux), pouls mensuel, récurrence des contributeurs (combien d'années chacun reste), battement horaire et carte de chaleur. Un détail compte : l'heure est celle de l'auteur, pas celle de l'analyste — `--date=format:` conserve le fuseau enregistré dans chaque commit, là où `format-local` réécrirait tout dans le fuseau de celui qui mesure.

```
#!/usr/bin/env python3
"""Demography and rhythm of a git repository, per year.
Usage: depot_demographie.py /path/to/clone"""
import subprocess, sys, collections
REPO = sys.argv[1]; SINCE = "2016-01-01"
def log(pretty, datefmt="short"):
    return subprocess.run(["git", "-C", REPO, "log", "--no-merges",
            f"--since={SINCE}", f"--date=format:{datefmt}",
            f"--pretty={pretty}"], capture_output=True,
    text=True).stdout.splitlines()

# 1) contributors per year, by tier (>=1, >=5, >=15 commits)
per = collections.defaultdict(collections.Counter)
for l in log("%ad|%aN|%aE", "%Y"):
    y, name, mail = l.split("|", 2)
    per[y][(name, mail.lower())] += 1
print("year  >=1  >=5  >=15  commits  median(core)")
for y in sorted(per):
    c = per[y]; core = sorted(n for n in c.values() if n >= 15)
    med = core[len(core)//2] if core else 0
    print(f"{y}  {len(c):4} {sum(1 for v in c.values() if v >= 5):4} "
          f"{len(core):4}  {sum(c.values()):6}  {med}")

```

```
# 2) recurrence: number of active years per contributor (2019+)
years = collections.defaultdict(set)
for l in log("%ad|%aN|%aE", "%Y"):
    y, name, mail = l.split("|", 2)
    if y >= "2019": years[(name, mail.lower())].add(y)
dist = collections.Counter(len(v) for v in years.values())
print("\nactive years -> contributors:", dict(sorted(dist.items())))

# 3) commits per month (the 10-year pulse)
mois = collections.Counter(log("%ad", "%Y-%m"))
print("\nper month:", dict(sorted(mois.items())))

# 4) heartbeat: hour + weekday in the AUTHOR's own timezone
#    (--date=format keeps each commit's recorded offset; format-local would
#     rewrite everything into the analyst's timezone — wrong for this use)
hours = collections.Counter(); days = collections.Counter()
for l in log("%ad", "%u %H"):
    d, h = l.split(); days[d] += 1; hours[h] += 1
print("\nby weekday (1=Mon):", dict(sorted(days.items())))
print("by hour:", dict(sorted(hours.items())))
```

La classification des commits : qui répare quoi

La méthode des graphes « Qui répare ? » et « Le mélange ne change pas » du chapitre « L'ambition non héroïque » : un commit est un correctif (fix, bug, regression, crash, segfault, leak, closes/fixes #N) ou une fonctionnalité (add, new, introduce, implement, support, allow, enable, initial), par mots-clés du message ; un correctif est une *rustine* — la taxe du travail récent — quand tous les fichiers `src/` qu'il touche avaient pour dernier modificateur le même auteur, à moins du seuil. Les seuils de 30, 90 et 180 jours donnent des résultats identiques sur Darktable ; le solde (refactorisation, nettoyage, build) est dit dans les légendes, car c'est le compartiment dominant. Les limites sont dites au même endroit du livre : la classification lit les messages, pas les diffs, et le dernier toucheur d'un fichier n'est pas toujours l'auteur du défaut — bruit symétrique, insensible au seuil.

```
#!/usr/bin/env python3
"""Commit classification: true repair / self-patch / feature / other.
A FIX is a SELF-patch when every src/ file it touches was last modified
by the same author less than THRESHOLD days earlier (30/90/180 give
identical results on Darktable).
Usage: depot_classification.py /path/to/clone [threshold_days]"""
import subprocess, sys, re, datetime, collections
REPO = sys.argv[1]; THR = int(sys.argv[2]) if len(sys.argv) > 2 else 90
FIX = re.compile(r"\b(fix|bug|regression|crash|segfau
                 r"|(closes|fixes)\s+#\d+", re.I)
FEAT = re.compile(r"\b(add|new|introduce|implement|su
                  r"|enable|initial)\b", re.I)
raw = subprocess.run(["git", "-C", REPO, "log", "--no-merges",
    "--since=2016-01-01", "--date=unix", "--pretty=%x1e%H|%at|%ae|%s",
    "--name-only", "--", "src/"], capture_output=True, text=True).stdout
# \x1e (record separator) as delimiter: emails contain "@", subjects "|"...
commits = []
for block in raw.split("\x1e")[1:]:
    lines = block.strip().splitlines()
```

```
    h, t, ae, s = lines[0].split("|", 3)
    commits.append((h, int(t), ae, s, [f for f in lines[1:] if f]))
commits.sort(key=lambda c: c[1])  # chronological
last = {}  # file -> (author, epoch)
stats = collections.defaultdict(collections.Counter)
for h, t, ae, s, files in commits:
    y = datetime.datetime.fromtimestamp(t, datetime.timezone.utc).year
    cls = "other"
    if FIX.search(s):
        prev = [last.get(f) for f in files]
        selfp = files and all(p and p[0] == ae and (t - p[1]) < THR*86400
                              for p in prev)
        cls = "self-fix" if selfp else "ext-fix"
    elif FEAT.search(s):
        cls = "feat"
    stats[y][cls] += 1; stats[y]["total"] += 1
    for f in files: last[f] = (ae, t)
print("year  total  ext-fix  self-fix  feat  (rest=other)")
for y in sorted(stats):
    c = stats[y]
    print(f"{y}  {c['total']:5}  {c['ext-fix']:6}  {c['self-fix']:7}  {c['feat']:5}")
```

Les pull requests : le veto et la concentration

La série du graphe « Le veto qui s'éteint » (« L'ambition non héroïque ») et la part du fusionneur principal (« Penser contre soi »). Deux choix de méthode font la différence avec un comptage naïf, et ils sont dans le script. Le fusionneur est le champ `merged_by` — la personne qui clique la fusion — et non l'auteur du commit de fusion : compter les commits de fusion gonfle les chiffres avec les fusions locales de branches et rend invisibles les fusions par rebasage. Et une proposition fermée sans fusion dont les commentaires disent « manually merged » ou « cherry-picked » est requalifiée en acceptation — 211 cas sur 1 583 fermetures, concentrés en 2018-2020, qui faisaient du pic de refus apparent de 2018 un artefact.

```
#!/usr/bin/env python3
"""Refusal rate + merger concentration, measured on pull requests.
Needs GITHUB_TOKEN. The merger = the merged_by field (who clicks the
merge), not merge-commit authorship. A closed-unmerged PR whose comments
say it was merged by hand is requalified as accepted."""
import os, requests, collections, re, time
API = "https://api.github.com/graphql"
H = {"Authorization": "bearer " + os.environ["GITHUB_TOKEN"]}
MANUAL = re.compile(r"manually merged|cherry.?picked"
                    r"|merged (this )?(by hand|manually)", re.I)
Q = """query($c:String) {repository(owner:"Darktable-org",name:"Darktable"){
 pullRequests(states:[MERGED,CLOSED],first:100,after:$c){
  pageInfo{hasNextPage endCursor}
  nodes{number state createdAt mergedAt mergedBy{login}
        comments(last:20){nodes{bodyText}}}}}"""
merged = collections.defaultdict(collections.Counter)
closed = collections.Counter(); requal = collections.Counter(); cur = None
while True:
```

```
    r = requests.post(API, json={"query": Q,
"variables": {"c": cur}},
                      headers=H).json()
    pr = r["data"]["repository"]
["pullRequests"]
    for n in pr["nodes"]:
        if n["state"] == "MERGED":
            who = n["mergedBy"]["login"] if
n["mergedBy"] else "?"
            merged[n["mergedAt"][:4]][who] +=
1
        else:
            y = n["createdAt"][:4]
            if
any(MANUAL.search(c["bodyText"])
                   for c in n["comments"]
["nodes"]):
                requal[y] += 1
            else:
                closed[y] += 1
    if not pr["pageInfo"]["hasNextPage"]:
break
    cur = pr["pageInfo"]["endCursor"];
time.sleep(0.5)
print("year  merged  refused  requalified
refusal%  top-merger-share")
for y in sorted(merged):
    m = sum(merged[y].values()); ref =
closed[y]
    top = max(merged[y].values()) if
merged[y] else 0
    print(f"{y}  {m:5}  {ref:5}
{requal[y]:3}  "
```

```
            f"{100*ref/(m+ref+requal[y]):5.1f}
    {100*top/m:5.1f}")
```

Les issues : la capacité de fermeture et les testeurs

Les séries de « Le déclin de la capacité de fermeture », « La fin de vie des issues » et « Les testeurs qui s'en vont » (« Livrer, pas larguer », « Le comité n'inventera rien ») : qui ferme — un humain ou le robot de péremption — se lit dans l'acteur de l'événement de fermeture ; les rapporteurs réguliers (≥ 3 issues dans l'année) et leur attrition se comptent sur les auteurs. La précaution du livre vaut pour tout ce qui suit : une fermeture n'est pas une réparation, et le script ne prétend pas trancher ce qu'un humain a fermé à raison.

```
#!/usr/bin/env python3
"""Issues: creation, closure capacity (human vs stale bot), reporters.
Needs GITHUB_TOKEN. The closer is the actor of the CLOSED_EVENT; the
stale bot answers to the github-actions login."""
import os, requests, collections, time
API = "https://api.github.com/graphql"; BOTS = {"github-actions"}
H = {"Authorization": "bearer " + os.environ["GITHUB_TOKEN"]}
Q = """query($c:String) {repository(owner:"Darktable-org",name:"Darktable"){
 issues(first:100,after:$c) {pageInfo{hasNextPage endCursor}
  nodes{createdAt closedAt author{login}
   timelineItems(itemTypes:[CLOSED_EVENT],last:1){nodes{
    ... on ClosedEvent{actor{login}}}}}}}}"""
created = collections.Counter(); human = collections.Counter()
bot = collections.Counter(); cur = None
authors = collections.defaultdict(collections.Counter)
while True:
    r = requests.post(API, json={"query": Q, "variables": {"c": cur}},
                      headers=H).json()
    ii = r["data"]["repository"]["issues"]
    for n in ii["nodes"]:
```

```
        y = n["createdAt"][:4]; created[y] += 1
        if n["author"]: authors[y][n["author"]["login"]] += 1
        if n["closedAt"]:
            ev = n["timelineItems"]["nodes"]
            actor = ev[0]["actor"]["login"] if ev and ev[0].get("actor") else "?"
            target = bot if actor in BOTS or actor.endswith("[bot]") else human
            target[n["closedAt"][:4]] += 1
    if not ii["pageInfo"]["hasNextPage"]: break
    cur = ii["pageInfo"]["endCursor"]; time.sleep(0.5)
print("year  created  closed-by-human  closed-by-bot  regulars(>=3)  gone")
prev = set()
for y in sorted(created):
    reg = {a for a, n in authors[y].items() if n >= 3}
    gone = len(prev - reg) if prev else ""
    print(f"{y}  {created[y]:5}  {human[y]:5}  {bot[y]:5}  {len(reg):4}  {gone}")
    prev = reg
```

Le forum : les fils, et comment on me nomme

Discourse expose tout en JSON — il suffit d'ajouter `.json` aux URL. Ce script compte les fils ouverts par an dans la catégorie Darktable de discuss.pixls.us (« Le découplage ») et les occurrences trimestrielles des formes de nommage (« Comment le forum me nomme », « Le cercle fermé »). La limite est dite dans la note du graphe : la recherche Discourse plafonne ses résultats ; les fenêtres trimestrielles gardent les comptes exacts pour des termes rares, et le total agrégé vient de `grouped_search_result`.

```
#!/usr/bin/env python3
"""discuss.pixls.us: threads per year in a category; naming forms per
quarter. Discourse serves JSON everywhere; no token needed, but paced."""
import requests, time, collections
BASE = "https://discuss.pixls.us"
def get(u):
    time.sleep(0.6)
    return requests.get(BASE + u, timeout=30).json()

# 1) threads opened per year in the Darktable category (id 19, /site.json)
topics = collections.Counter(); page = 0
while True:
    d = get(f"/c/Darktable/19.json?page={page}")
    tl = d["topic_list"]["topics"]
    if not tl: break
    for t in tl: topics[t["created_at"][:4]] += 1
    page += 1
print("threads/year:", dict(sorted(topics.items())))

# 2) naming study: occurrences of each form, per quarter
for term in ('"Aurélien"', '"AP"'):
    for y in range(2019, 2027):
        for q, (a, b) in enumerate([("01-01", "04-01"), ("04-01", "07-01"),
```

```
                                        ("07-01",
"10-01"), ("10-01", "12-31")], 1):
            d = get(f"/search.json?q=
{term}%20after:{y}-{a}%20before:{y}-{b}")
            n =
d.get("grouped_search_result",
{}).get("post_count",
                    len(d.get("posts",
[])))
            print(f"{term} {y}-Q{q}: {n}")
```

Le salon Matrix : le comptage sur l'export

L'export du salon est privé — c'est la seule source du livre que le lecteur ne peut pas re-mesurer, et les notes qui s'y appuient le disent à chaque fois. Le script de comptage est publié par honnêteté d'instrument : il dit exactement ce qui a été compté (messages de type `m.room.message`, par expéditeur, par trimestre). Les codages fins — distinguer l'entraide adressée du bavardage — ont été faits à la main sur cette sortie, et sont décrits dans les notes qui les utilisent.

```python
#!/usr/bin/env python3
"""Element/Matrix room export (JSON): messages per sender, per quarter.
The export is PRIVATE (names live as roles in the book); this script
documents the counting, it cannot re-fetch the data.
Usage: salon_matrix.py export.json [sender_id]"""
import json, sys, datetime, collections
d = json.load(open(sys.argv[1]))
who = sys.argv[2] if len(sys.argv) > 2 else None
per = collections.Counter(); senders = collections.Counter()
for m in d["messages"]:
    if m.get("type") != "m.room.message": continue
    t = datetime.datetime.fromtimestamp(m["origin_serv

datetime.timezone.utc)
    key = f"{t.year}-Q{(t.month-1)//3+1}"
    senders[m["sender"]] += 1
    if not who or m["sender"] == who: per[key] += 1
print("top senders:", senders.most_common(10))
print("per quarter:", dict(sorted(per.items())))
```

Des séries aux graphes

Les graphes du livre sont des fichiers JSON au format Plotly (« data » + « layout »), lus par un shortcode Hugo qui porte le titre et la légende. Le gabarit :

```python
#!/usr/bin/env python3
"""Series -> the book's chart JSON ({"data":[...],"layout":{...}}),
read by the Hugo plotly shortcode. Title and caption live in the
shortcode call, not in the JSON."""
import json
def chart(series, xtitle="", ytitle="", kind="scatter", out="chart.json"):
    data = [{"type": kind, "name": name, "x": x, "y": y}
            for name, (x, y) in series.items()]
    layout = {"xaxis": {"title": {"text": xtitle}},
              "yaxis": {"title": {"text": ytitle}},
              "margin": {"l": 60, "r": 20, "t": 10, "b": 40},
              "legend": {"orientation": "h"}}
    json.dump({"data": data, "layout": layout}, open(out, "w"),
              ensure_ascii=False, indent=1)
if __name__ == "__main__":
    chart({"demo": ([2019, 2020, 2021], [1, 4, 2])}, "année", "valeur")
```

Les trois graphes de Brooks (« Le comité n'inventera rien ») ne minent rien : ce sont des courbes analytiques — n(n−1)/2 canaux pour n personnes — tracées avec ce gabarit. Le graphe de l'épuisement (« Le travailleur invisible ») combine six séries personnelles issues des scripts ci-dessus — commits, propositions

ouvertes, relectures, tri des issues, messages sur les deux forums — chacune normalisée à son propre sommet ; et le graphe « La maison vide » croise trois sources (issues GitHub, contributeurs du dépôt, entraide du salon) avec les scripts correspondants, en excluant mes propres contributions là où la légende le dit.

Voilà les instruments. Ils tiennent en trois cents lignes, et c'est le point : aucune mesure de ce livre ne demande un laboratoire. Elles demandaient seulement qu'on regarde les registres que le projet tient lui-même, publiquement, depuis dix ans — et qu'on accepte d'écrire ce qu'on y trouve.

Table des matières

Prologue — Trois chemins

Partie I — Ce que concevoir veut dire

1. Le mot et la chose
2. Il n'y a pas de problème technique
3. Écouter sans obéir

Partie II — La discipline

4. La racine et le symptôme
5. Poser le problème
6. Mesurer sans se mentir
7. Compter les absents
8. Décider, puis tenir
9. S'outiller sans se soumettre

Partie III — La machine

10. Sous l'interface, les dragons
11. L'éloge du simple
12. Le code est un passif
13. Le culte du cargo
14. L'intelligence en amont
15. L'outil transparent
16. L'instrument et la machine
17. Ouvrir sans tenir le pinceau

18. Finir et maintenir

Partie IV — Le collectif

19. Livrer, pas larguer
20. Le comité n'inventera rien
21. Le travailleur invisible
22. Travailler pour le travail
23. L'ambition non héroïque

Partie V — La liberté et ses conditions

24. La liberté dans le vide
25. Ce que l'entre-soi produit
26. La violence à trois étages
27. Le cercle fermé
28. Le conflit bien tempéré
29. Penser contre soi
30. Le travailleur incompétent
31. Le pacte

Conclusion — Prendre soin

Épilogue — Derrière chaque problème, un humain

Annexe — Les instruments de mesure